

# S3F(C)8285X/9X/BX

---

8位CMOS单片机

Revision 1.50

April 2010

## 用户手册

SAMSUNG ELECTRONICS RESERVES THE RIGHT TO CHANGE PRODUCTS, INFORMATION AND SPECIFICATIONS WITHOUT NOTICE.

Products and specifications discussed herein are for reference purposes only. All information discussed herein is provided on an "AS IS" basis, without warranties of any kind.

This document and all information discussed herein remain the sole and exclusive property of Samsung Electronics. No license of any patent, copyright, mask work, trademark or any other intellectual property right is granted by one party to the other party under this document, by implication, estoppel or otherwise.

Samsung products are not intended for use in life support, critical care, medical, safety equipment, or similar applications where product failure could result in loss of life or personal or physical harm, or any military or defense application, or any governmental procurement to which special terms or provisions may apply.

For updates or additional information about Samsung products, contact your nearest Samsung office.

All brand names, trademarks and registered trademarks belong to their respective owners.

© 2010 Samsung Electronics Co., Ltd. All rights reserved.

# Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

**S3F(C)8285X/9X/BX 8位CMOS单片机**  
用户手册, Revision 1.50

**Copyright © 2010 Samsung Electronics Co., Ltd.**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Samsung Electronics.

Samsung Electronics Co., Ltd.  
San #24 Nongseo-Dong, Giheung-Gu  
Yongin-City, Gyeonggi-Do, Korea 446-711

TEL : (82)-(31)-209-3865  
FAX : (82)-(31)-209-6494

Home Page: <http://www.samsungsemi.com>

Printed in the Republic of Korea

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product.

# Revision History

Revision No.	Date	Description	Author(s)
1.50	Apr. 02, 2010	- Initial draft	younghee.won

# Table of Contents

<b>1</b>	<b>产品概述</b>	<b>1-1</b>
1.1	S3C8-系列MCU	1-1
1.2	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X MCU	1-2
1.2.1	Flash	1-2
1.2.2	特性	1-3
1.2.2.1	CPU	1-3
1.2.2.2	存储器	1-3
1.2.2.3	指令及	1-3
1.2.2.4	65 I/O 口	1-3
1.2.2.5	中断	1-4
1.2.2.6	8位 Basic Timer	1-4
1.2.2.7	8位 Timer/Counter A	1-4
1.2.2.8	8位 Timer/Counter B	1-4
1.2.2.9	16位 Timer/Counter 0	1-4
1.2.2.10	16位 Timer/Counter 1	1-4
1.2.2.11	钟表定时器 (Watch Timer)	1-4
1.2.2.12	LCD 控制/驱动器	1-4
1.2.2.13	A/D 转换	1-5
1.2.2.14	UART	1-5
1.2.2.15	8位串行输入输出接口	1-5
1.2.2.16	电池电压检测	1-5
1.2.2.17	低电压复位 (LVR)	1-5
1.2.2.18	2 种低功耗模式	1-5
1.2.2.19	振荡源	1-5
1.2.2.20	指令执行时间	1-5
1.2.2.21	工作电压范围	1-6
1.2.2.22	工作温度范围	1-6
1.2.2.23	封装形式	1-6
1.2.2.24	Smart Option	1-6
1.3	内部模块框图	1-7
1.4	管脚分布图	1-8
1.5	管脚特性描述	1-10
1.6	管脚电路	1-12

<b>2</b>	<b>地址空间</b>	<b>2-1</b>
2.1	概述	2-1
2.2	程序存储空间 (ROM)	2-2
2.3	Smart Option	2-3
2.4	寄存器结构	2-5
2.5	寄存器页面指针(PP)	2-10
2.5.1	寄存器SET 1	2-12
2.5.2	寄存器SET 2	2-12
2.5.3	主寄存器区	2-13
2.5.4	工作寄存器	2-14
2.5.5	使用寄存器指针	2-15
2.6	寄存器寻址	2-17
2.6.1	通用工作寄存器区 (C0H–CFH)	2-19
2.6.2	4 位工作寄存器寻址方式	2-20
2.6.3	8 位工作寄存器寻址方式	2-22
2.7	系统和用户栈	2-24
2.7.1	栈操作	2-24
2.7.2	用户自定义栈	2-24
2.7.3	栈指针 (SPL, SPH)	2-24
<b>3</b>	<b>寻址模式</b>	<b>3-1</b>
3.1	概述	3-1
3.2	寄存器寻址模式(R)	3-2
3.3	间接寄存器寻址模式(IR)	3-3
3.4	偏址寻址模式(X)	3-7
3.5	直接寻址模式(DA)	3-10
3.6	间接寻址模式(IA)	3-12
3.7	相对地址寻址模式(RA)	3-13
3.8	立即数寻址模式(IM)	3-14
<b>4</b>	<b>控制寄存器</b>	<b>4-1</b>
4.1	概述	4-1
4.1.1	ADCON — A/D 转换控制寄存器: F3H - Set 1, Bank0	4-5
4.1.2	BLDCON — 电池电压检测控制寄存器: D2H - Set 1, Bank0	4-6
4.1.3	BTCON — Basic Timer 控制寄存器: D3H - Set 1	4-7
4.1.4	CLKCON — 系统控制寄存器: D4H - Set 1	4-8
4.1.5	FLAGS — 系统标志寄存器: D5H - Set 1	4-9
4.1.6	FMCON — 闪存 控制寄存器: D2H - Set 1, Bank1	4-10
4.1.7	FMSECH — 闪存扇区地址寄存器 (高字节): D0H - Set 1, Bank1	4-11
4.1.8	FMSECL — 闪存扇区地址寄存器 (低字节): D1H - Set 1, Bank1	4-11
4.1.9	FMUSR — 闪存用户可编程使能寄存器: FFH - Set 1, Bank1	4-12
4.1.10	IMR — 中断屏蔽寄存器: DDH - Set 1	4-13
4.1.11	INTPND — 中断标志寄存器: F9H - Set 1, Bank0	4-15
4.1.12	IPH — 指令指针 (High Byte): DAH - Set 1	4-16
4.1.13	IPL — 指令指针(低字节): DBH - Set 1	4-16
4.1.14	IPR — 中断优先级寄存器: FFH - Set 1, Bank0	4-17
4.1.15	IRQ — 中断请求寄存器: DCH - Set 1	4-18

4.1.16 LCON — LCD 控制寄存器: D0H - Set 1, Bank0.....	4-20
4.1.17 OSCCON — 时钟 控制寄存器: FAH - Set 1, Bank0 .....	4-21
4.1.18 P0CONH — P0 控制寄存器 (高字节): E0H - Set 1, Bank 1.....	4-22
4.1.19 P0CONL — P0 口控制寄存器 (低字节): E1H - Set 1, Bank 1.....	4-23
4.1.20 P0INTH — P0 口中断控制寄存器 (高字节): E2H - Set 1, Bank1 .....	4-24
4.1.21 P0INTL — P0 中断控制寄存器 (低字节): E3H - Set 1, Bank1 .....	4-25
4.1.22 P0PND — P0 中断标志寄存器: E4H - Set 1, Bank1.....	4-26
4.1.23 P1CONH — P1 控制寄存器 (高字节): E5H - Set 1, Bank1.....	4-27
4.1.24 P1CONL — P1 口控制寄存器 (低字节): E6H - Set 1, Bank1 .....	4-28
4.1.25 P1PUR — P1 口上拉电阻使能控制寄存器: E7H - Set 1, Bank1 .....	4-29
4.1.26 P2CONH — P2 控制寄存器 (高字节): E8H - Set 1, Bank1.....	4-30
4.1.27 P2CONL — P2 口控制寄存器 (低字节): E9H - Set 1, Bank1 .....	4-31
4.1.28 P3CONH — P3 口控制寄存器 (高字节): EAH - Set 1, Bank1.....	4-32
4.1.29 P3CONL — P3 口控制寄存器 (低字节): EBH - Set 1, Bank1 .....	4-33
4.1.30 P4CONH — P4 控制寄存器 (高字节): ECH - Set 1, Bank1 .....	4-34
4.1.31 P4CONL — P4 口控制寄存器 (低字节): EDH - Set 1, Bank1 .....	4-35
4.1.32 P4PUR — P4 口上拉电阻使能位: EEH - Set 1, Bank1.....	4-36
4.1.33 P5CONH — P5 口控制寄存器 (高字节): F9H - Set 1, Bank1 .....	4-38
4.1.34 P5CONL — P5 口控制寄存器 (低字节): FAH - Set 1, Bank1 .....	4-39
4.1.35 P5PUR — P5 口上拉电阻使能控制寄存器: EFH - Set 1, Bank1 .....	4-40
4.1.36 P6CONH — P6 口控制寄存器 (高字节): FBH - Set 1, Bank1 .....	4-42
4.1.37 P6CONL — P6 口控制寄存器(低字节): FCH- Set 1, Bank1 .....	4-43
4.1.38 P7CON — P7 口控制寄存器: FDH - Set 1, Bank1 .....	4-44
4.1.39 P8CON — P8 口控制寄存器: FEH - Set 1, Bank1.....	4-45
4.1.40 PP — 寄存器页指针: DFH - Set 1.....	4-46
4.1.41 RP0 — 寄存器指针 0: D6H - Set 1 .....	4-48
4.1.42 RP1 — 寄存器指针 1: D7H - Set 1.....	4-48
4.1.43 SIOCON — SIO 控制寄存器: E0H - Set 1, Bank0 .....	4-49
4.1.44 SPH — 堆栈指针 (高字节): D8H - Set 1 .....	4-50
4.1.45 SPL — 堆栈指针 (低字节): D9H - Set 1.....	4-50
4.1.46 STPCON — Stop 控制寄存器: FBH - Set 1, Bank 0 .....	4-51
4.1.47 SYM — 系统模式寄存器: DEH - Set 1.....	4-52
4.1.48 T0CON — Timer 0 控制寄存器: E3H - Set 1, Bank0.....	4-53
4.1.49 T1CON — Timer 1控制寄存器: EBH - Set 1, Bank0 .....	4-54
4.1.50 TACON — Timer A 控制寄存器: E8H - Set 1, Bank0.....	4-55
4.1.51 TBCON — Timer B 控制寄存器: F2H - Set 1, Bank0 .....	4-56
4.1.52 UARTCON — UART 控制寄存器: F6H - Set 1, Bank0.....	4-57
4.1.53 WTCN — 钟表定时器控制寄存器: D1H - Set 1, Bank0 .....	4-59

## 5 中断结构 .....5-1

5.1 概述 .....	5-1
5.1.1 中断级 (Levels).....	5-1
5.1.2 中断向量 (Vectors).....	5-1
5.1.3 中断源 (Sources).....	5-1
5.1.4 中断类型.....	5-2
5.1.5 S3C828BX/C8289X/C8285X 中断结构.....	5-3
5.1.6 中断向量地址 .....	5-5
5.1.7 使能/禁止中断的指令 (EI, DI).....	5-7
5.1.8 系统级中断控制寄存器.....	5-7
5.1.9 中断处理控制要点 .....	5-8
5.1.10 外设中断控制寄存器 .....	5-9
5.1.11 系统模式寄存器 (SYM).....	5-10
5.1.12 中断屏蔽寄存器 (IMR).....	5-11
5.1.13 中断优先级寄存器 (IPR).....	5-12
5.1.14 中断请求寄存器 (IRQ).....	5-14
5.1.15 中断标志位类型.....	5-15
5.1.15.1 概述 .....	5-15
5.1.15.2 硬件自动清零的标志位 .....	5-15
5.1.15.3 中断服务程序中清零的标志位 .....	5-15
5.1.16 中断源查询顺序.....	5-16
5.1.17 中断服务程序 .....	5-16
5.1.18 中断向量地址的生成 .....	5-17
5.1.19 中断嵌套.....	5-17
5.1.20 指令指针 (IP).....	5-17
5.1.21 快速中断处理 .....	5-18
5.1.21.1 快速中断初始化步骤 .....	5-18
5.1.21.2 快速中断服务程序.....	5-18
5.1.21.3 中断标志位类型间关系 .....	5-19
5.1.21.4 编程指导 .....	5-19

## 6 指令集 .....6-1

6.1 概述 .....	6-1
6.1.1 数据类型.....	6-1
6.1.2 寄存器访问 .....	6-1
6.1.3 寻址模式.....	6-1
6.1.4 标志寄存器(FLAGS).....	6-5
6.1.5 标志位描述 .....	6-6
6.1.5.1 C: 进(借)位标志 (FLAGS.7) .....	6-6
6.1.5.2 Z: 零标志位(FLAGS.6).....	6-6
6.1.5.3 S: 符号标志位 (FLAGS.5).....	6-6
6.1.5.4 V: 溢出标志 (FLAGS.4).....	6-6
6.1.5.5 D: 十进制调整标志 (FLAGS.3) .....	6-6
6.1.5.6 H: 半字节进(借)位标志 (FLAGS.2).....	6-6
6.1.5.7 FIS: 快速中断状态标志 (FLAGS.1).....	6-6
6.1.5.8 BA: 寄存器块地址标志 (FLAGS.0).....	6-6

6.1.6 指令集符号 .....	6-7
6.1.7 条件码 .....	6-11
6.1.8 指令集描述 .....	6-12
6.1.8.1 ADC — 带进位加法 (Add with Carry) .....	6-13
6.1.8.2 ADD — 加法 (Add) .....	6-14
6.1.8.3 AND — 逻辑与 (Logical AND) .....	6-15
6.1.8.4 BAND — 位与 (Bit AND) .....	6-16
6.1.8.5 BCP — 位比较 (Bit Compare) .....	6-17
6.1.8.6 BITC — 位反 (Bit Complement) .....	6-18
6.1.8.7 BITR — 位清零 (Bit Reset) .....	6-19
6.1.8.8 BITS — 位置1 (Bit Set) .....	6-20
6.1.8.9 BOR — 位或 (Bit OR) .....	6-21
6.1.8.10 BTJRF — 位测试, 若为假相对跳转 (Bit Test, Jump Relative on False) .....	6-22
6.1.8.11 BTJRT — 位测试, 若为真, 相对跳转 (Bit Test, Jump Relative on True) .....	6-23
6.1.8.12 BXOR — 位异或 (Bit XOR) .....	6-24
6.1.8.13 CALL — 程序调用 (Call Procedure) .....	6-25
6.1.8.14 CCF — 进位标志位取反 (Complement Carry Flag) .....	6-26
6.1.8.15 CLR — 清零 (Clear) .....	6-27
6.1.8.16 COM — 取反 (complement) .....	6-28
6.1.8.17 CP — 比较 (Compare) .....	6-29
6.1.8.18 CPIJE — 比较, 增加一, 若相等跳转 (Compare, Increment, and Jump on Equal) .....	6-30
6.1.8.19 CPIJNE — 比较, 增加一, 不等跳转 (Compare, Increment, Jump on Non-Equal) .....	6-31
6.1.8.20 DA — 十进制调整 (Decimal Adjust) .....	6-32
6.1.8.21 DEC — 字节减1 (Decrement) .....	6-34
6.1.8.22 DECW — 字减1 (Decrement Word) .....	6-35
6.1.8.23 DI — 屏蔽全局中断 (Disable Interrupts) .....	6-36
6.1.8.24 DIV — 无符号除法 (Unsigned Divide) .....	6-37
6.1.8.25 DJNZ — 减1, 如果非零, 跳转 (Decrement and Jump if Non-Zero) .....	6-38
6.1.8.26 EI — 使能全局中断 (Enable Interrupts) .....	6-39
6.1.8.27 ENTER — 进入 (Enter) .....	6-40
6.1.8.28 EXIT — 退出 (Exit) .....	6-41
6.1.8.29 IDLE — 空闲指令 (Idle Operation) .....	6-42
6.1.8.30 INC — 加1 (Increment) .....	6-43
6.1.8.31 INCW — 字加1 (Increment Word) .....	6-44
6.1.8.32 IRET — 中断返回 (Interrupt Return) .....	6-45
6.1.8.33 JP — 跳转 (Jump) .....	6-46
6.1.8.34 JR — 相对跳转指令 (Jump Relative) .....	6-47
6.1.8.35 LD — 传送数据 (Load) .....	6-48
6.1.8.36 LDB — 传送位数据 (Load Bit) .....	6-50
6.1.8.37 LDC/LDE — 传送程序/外部数据存储器数据 (Load Memory) .....	6-51
6.1.8.38 LDCD/LDED — 传送数据之后地址减1 (Load Memory and Decrement) .....	6-53
6.1.8.39 LDCI/LDEI — 传送数据后地址加1 (Load Memory and Increment) .....	6-54
6.1.8.40 LDCPD/LDEPD — 传送数据前地址减1 (Load Memory with Pre-Decrement) .....	6-55
6.1.8.41 LDCPI/LDEPI — 传送数据前地址加1 (Load Memory with Pre-Increment) .....	6-56
6.1.8.42 LDW — 传送字数据 (Load Word) .....	6-57
6.1.8.43 MULT — 无符号数乘法 (Unsigned Multiply) .....	6-58
6.1.8.44 NEXT — Next 指令 .....	6-59
6.1.8.45 NOP — 空操作 (No Operation) .....	6-60
6.1.8.46 OR — 逻辑或 (Logical OR) .....	6-61



6.1.8.47 POP — 出栈 (Pop from Stack).....	6-62
6.1.8.48 POPUD — 弹出用户栈 (自减) (Pop User Stack (Decrementing)).....	6-63
6.1.8.49 POPUI — 弹出用户栈 (自增) (Pop User Stack (Incrementing)).....	6-64
6.1.8.50 PUSH — 压栈 (Push to Stack).....	6-65
6.1.8.51 PUSHUD — 压入用户栈 (自减) Push User Stack (Decrementing).....	6-66
6.1.8.52 PUSHUI — 压入用户栈 (自增) Push User Stack (Incrementing).....	6-67
6.1.8.53 RCF — C清零 (Reset Carry Flag).....	6-68
6.1.8.54 RET — 子程序返回 (Return).....	6-69
6.1.8.55 RL — 左移 (Rotate Left).....	6-70
6.1.8.56 RLC — 带进位左移 (Rotate Left Through Carry).....	6-71
6.1.8.57 RR — 右移 (Rotate Right).....	6-72
6.1.8.58 RRC — 带进位右移 (Rotate Right Through Carry).....	6-73
6.1.8.59 SB0 — 选择 Bank 0 (Select Bank 0).....	6-74
6.1.8.60 SB1 — 选择 Bank 1 (Select Bank 1).....	6-75
6.1.8.61 SBC — 带进位减法 (Subtract with Carry).....	6-76
6.1.8.62 SCF — C置1 (Set Carry Flag).....	6-77
6.1.8.63 SRA — 算术右移 (Shift Right Arithmetic).....	6-78
6.1.8.64 SRP/SRP0/SRP1 — 设置寄存器指针 (Set Register Pointer).....	6-79
6.1.8.65 STOP — Stop 操作 (Stop Operation).....	6-80
6.1.8.66 SUB — 减法 (Subtract).....	6-81
6.1.8.67 SWAP — 交换 (Swap Nibbles).....	6-82
6.1.8.68 TCM — 取反后位测试 (Test Complement Under Mask).....	6-83
6.1.8.69 TM — 位测试 (Test Under Mask).....	6-84
6.1.8.70 WFI — 等待中断 (Wait for Interrupt).....	6-85
6.1.8.71 XOR — 逻辑异或 (Logical Exclusive OR).....	6-86

## 7 时钟电路.....7-1

7.1 概述.....	7-1
7.1.1 系统时钟电路.....	7-1
7.1.2 CPU 时钟助记符.....	7-1
7.1.3 主振荡器电路.....	7-2
7.1.4 副振荡器电路.....	7-3
7.1.5 省电模式下时钟状态.....	7-4
7.1.6 系统时钟控制寄存器 (CLKCON).....	7-5
7.1.7 CPU时钟切换.....	7-7

## 8 复位和省电模式.....8-1

8.1 系统复位.....	8-1
8.1.1 概述.....	8-1
8.1.2 正常模式复位操作.....	8-1
8.1.3 硬件复位值.....	8-2
8.2 省电模式.....	8-5
8.2.1 Stop 模式.....	8-5
8.2.2 使用 nRESET 退出 STOP 模式.....	8-5
8.2.3 使用外部中断退出 STOP 模式.....	8-5
8.2.4 使用内部中断退出STOP 模式.....	8-5
8.2.5 如何进入STOP 模式.....	8-6
8.2.6 Idle 模式.....	8-6

<b>9</b>	<b>I/O 口 .....</b>	<b>9-1</b>
9.1	概述 .....	9-1
9.1.1	口数据寄存器 .....	9-2
9.2	P0 口 .....	9-3
9.2.1	P0 口控制寄存器 (P0CONH, P0CONL) .....	9-3
9.2.2	P0 口中断使能和标志位寄存器 (P0INTH, P0INTL) .....	9-3
9.3	P1 口 .....	9-7
9.3.1	P1 口控制寄存器 (P1CONH, P1CONL) .....	9-7
9.3.2	P1 口上拉电阻使能寄存器 (P1PUR) .....	9-7
9.4	P2 口 .....	9-9
9.4.1	P2 口控制寄存器 (P2CONH, P2CONL) .....	9-9
9.5	P3 口 .....	9-11
9.5.1	P3 口控制寄存器 (P3CONH, P3CONL) .....	9-11
9.6	P4 口 .....	9-13
9.6.1	P4 口控制寄存器 (P4CONH, P4CONL) .....	9-13
9.6.2	P4 口上拉电阻使能寄存器 (P4PUR) .....	9-13
9.7	P5 口 .....	9-15
9.7.1	P5 口控制寄存器 (P5CONH, P5CONL) .....	9-15
9.7.2	P5 口上拉电阻使能寄存器 (P5PUR) .....	9-15
9.8	P6 口 .....	9-17
9.8.1	P6 口控制寄存器 (P6CONH, P6CONL) .....	9-17
9.9	P7 口 .....	9-19
9.9.1	P7 口控制寄存器 (P7CON) .....	9-19
9.10	P8 口 .....	9-20
9.10.1	P8 口控制寄存器 (P8CON) .....	9-20
<b>10</b>	<b>Basic Timer .....</b>	<b>10-1</b>
10.1	概述 .....	10-1
10.1.1	Basic Timer (BT) .....	10-1
10.1.2	Basic Timer控制寄存器 (BTCON) .....	10-2
10.1.3	Basic Timer功能描述 .....	10-3
10.1.3.1	看门狗定时器功能 .....	10-3
10.1.3.2	振荡稳定功能 .....	10-3

<b>11</b>	<b>8位timer A/B</b>	<b>11-1</b>
11.1	8位timer A	11-1
11.1.1	概述	11-1
11.1.2	Timer A 控制寄存器 (TACON)	11-2
11.1.3	Timer A 功能描述	11-3
11.1.3.1	Timer A 中断 (IRQ0, 中断向量地址: DCH 和 DEH)	11-3
11.1.3.2	Interval (定时) 模式	11-3
11.1.4	PWM 模式	11-4
11.1.5	Capture (捕获) 模式	11-5
11.1.6	模块框图	11-6
11.2	8位 Timer B	11-7
11.2.1	概述	11-7
11.2.2	模块框图	11-8
11.2.3	Timer b 脉冲宽度计算	11-9
<b>12</b>	<b>16位 Timer 0/1</b>	<b>12-1</b>
12.1	16位timer 0	12-1
12.1.1	概述	12-1
12.2	功能描述	12-1
12.2.1	定时模式	12-1
12.2.2	Timer 0 控制寄存器 (T0CON)	12-2
12.2.3	框图	12-3
12.3	16位Timer 1	12-4
12.3.1	概述	12-4
12.3.2	Timer 1 控制寄存器 (T1CON)	12-5
12.3.3	Timer 1 功能描述	12-6
12.3.3.1	Timer 1 中断 (IRQ2, 中断向量地址 E4H 和 E6H)	12-6
12.3.4	定时模式	12-6
12.3.5	PWM 模式	12-7
12.3.6	捕获模式	12-8
12.3.7	框图	12-9
<b>13</b>	<b>钟表定时器</b>	<b>13-1</b>
13.1	概述	13-1
13.1.1	钟表定时器控制寄存器 (WTCON: R/W)	13-2
13.1.2	钟表定时器电路图	13-3
<b>14</b>	<b>LCD 控制器/驱动器</b>	<b>14-1</b>
14.1	概述	14-1
14.1.1	LCD 电路框图	14-2
14.1.2	LCD RAM 地址空间	14-3
14.1.3	LCD 控制寄存器 (LCON)	14-4
14.1.4	LCD 分压电阻	14-5
14.1.5	Common (COM) 信号	14-6
14.1.6	Segment (SEG) 信号	14-6

<b>15</b>	<b>10 位 AD 转换器 .....</b>	<b>15-1</b>
15.1	概述 .....	15-1
15.1.1	功能描述 .....	15-1
15.1.2	转换时间 .....	15-2
15.1.3	A/D 转换控制寄存器 (ADCON) .....	15-2
15.1.4	内部参考电压 .....	15-3
15.1.5	模块图 .....	15-4
<b>16</b>	<b>串行输入输出接口 .....</b>	<b>16-1</b>
16.1	概述 .....	16-1
16.1.1	编程流程 .....	16-1
16.1.2	SIO控制寄存器 (SIOCON) .....	16-2
16.1.3	SIO 预分频寄存器 (SIOPS) .....	16-3
16.1.4	模块框图 .....	16-3
16.1.5	SIO 时序图 .....	16-4
<b>17</b>	<b>UART .....</b>	<b>17-1</b>
17.1	概述 .....	17-1
17.1.1	编程流程 .....	17-1
17.1.2	UART控制寄存器(UARTCON) .....	17-2
17.1.3	UART 中断标志位 .....	17-3
17.1.4	UART 数据寄存器 (UDATA) .....	17-4
17.1.5	UART 波特率数据寄存器 (BRDATA) .....	17-4
17.1.6	波特率计算 .....	17-4
17.1.6.1	Mode 0 模式下波特率的计算 .....	17-4
17.1.6.2	Mode 2 模式下波特率的计算 .....	17-4
17.1.6.3	Mode 1 和mode 3 模式下波特率的计算 .....	17-4
17.1.7	模块框图 .....	17-6
17.1.8	UART Mode 0 模式功能描述 .....	17-7
17.1.8.1	Mode 0 模式数据发送流程: .....	17-7
17.1.8.2	Mode 0 模式数据接收流程: .....	17-7
17.1.9	UART Mode 1 模式功能描述 .....	17-9
17.1.9.1	Mode 1 模式数据发送流程 .....	17-9
17.1.9.2	Mode 1 模式数据接收流程 .....	17-9
17.1.10	UART Mode 2 模式功能描述 .....	17-10
17.1.10.1	Mode 2 模式发送数据流程 .....	17-10
17.1.10.2	Mode 2 模式数据接收流程 .....	17-10
17.1.11	UART Mode 3 模式功能描述 .....	17-11
17.1.11.1	Mode 3 模式发送数据流程 .....	17-11
17.1.11.2	Mode 3 模式数据接收流程 .....	17-11
17.1.12	多节点串行通讯 .....	17-12
17.1.13	主机/从机交互协议的一个例子 .....	17-12
17.1.14	多节点通讯的建立流程 .....	17-13

<b>18</b>	<b>电池电压检测</b>	<b>18-1</b>
18.1	概述	18-1
18.1.1	电池电压检测控制寄存器 (BLDCON)	18-2
<b>19</b>	<b>嵌入式闪存接口</b>	<b>19-1</b>
19.1	概述	19-1
19.2	用户编程模式	19-2
19.2.1	闪存控制寄存器 (用户编程模式)	19-2
19.2.2	闪存用户编程使能寄存器 (FMUSR)	19-3
19.2.3	闪存扇区地址寄存器	19-4
19.3	ISP™ (在板编程) 扇区	19-5
19.3.1	ISP复位向量和 ISP 扇区大小的关系	19-6
19.3.2	扇区擦除	19-7
19.3.3	用户编程模式下扇区擦除流程	19-8
19.3.4	编程	19-9
19.3.4.1	用户编程模式下的编程流程	19-9
19.3.5	读	19-11
19.4	Hard Lock 保护	19-12
<b>20</b>	<b>电气参数</b>	<b>20-1</b>
20.1	概述	20-1
<b>21</b>	<b>机械尺寸</b>	<b>21-1</b>
21.1	概述	21-1
<b>22</b>	<b>S3F828BX/F8289X/F8285X Flash MCU</b>	<b>22-1</b>
22.1	概述	22-1
22.1.1	编程工作模式特性	22-5
<b>23</b>	<b>开发工具</b>	<b>23-1</b>
23.1	概述	23-1
23.1.1	Shine	23-1
23.1.2	SAMA 编译器	23-1
23.1.3	SASM88	23-1
23.1.4	HEX2ROM	23-1
23.1.5	目标板	23-1
23.1.6	TB828B/9/5 目标板	23-3
23.1.7	SMDS2+ 选择 (SAM8)	23-5
23.1.7.1	IDLE LED	23-5
23.1.7.2	STOP LED	23-5

# List of Figures

Figure Number	Title	Page Number
图 1-1	内部模块框图 .....	1-7
图 1-2	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 管脚分布 (80-QFP-1420C) .....	1-8
图 1-3	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X管脚分布(80-TQFP-1212) .....	1-9
图 1-4	管脚电路类型 A .....	1-12
图 1-5	管脚电路类型B .....	1-12
图 1-6	管脚电路类型C .....	1-12
图 1-7	管脚电路类型D-1 (P3.4, P3.5) .....	1-13
图 1-8	管脚电路类型E-4 (P0, P1) .....	1-13
图 1-9	管脚电路类型F-1 (P2.0-P2.6) .....	1-14
图 1-10	管脚电路类型F-2 (P2.7) .....	1-14
图 1-11	管脚电路类型H-4 .....	1-15
图 1-12	管脚电路类型H-8 (P4, P5) .....	1-15
图 1-13	管脚电路类型H-9 (P3.0-P3.3, P6, P7, P8) .....	1-16
图 2-1	程序存储地址空间 .....	2-2
图 2-2	Smart Option .....	2-3
图 2-3	内部寄存器卷的地址空间 (S3C828BX/F828BX) .....	2-7
图 2-4	内部寄存器卷的地址空间(S3C8289X/F8289X) .....	2-8
图 2-5	内部寄存器卷的地址空间 (S3C8285X/F8285X) .....	2-9
图 2-6	寄存器页面指针 (PP) .....	2-10
图 2-7	Set 1, Set 2, 主寄存器区和 LCD 数据寄存器 .....	2-13
图 2-8	8字节工作寄存器区 (Slices) .....	2-14
图 2-9	相邻的16字节工作寄存器块 .....	2-15
图 2-10	非相邻的 16 字节工作寄存器块 .....	2-16
图 2-11	16位工作寄存器结构 .....	2-17
图 2-12	寄存器卷寻址模式 .....	2-18
图 2-13	通用工作寄存器区 .....	2-19
图 2-14	4位工作寄存器寻址方式 .....	2-21
图 2-15	4位工作寄存器寻址方式实例 .....	2-21
图 2-16	8位工作寄存器寻址方式 .....	2-22
图 2-17	8位工作寄存器寻址方式实例 .....	2-23
图 2-18	栈操作 .....	2-24
图 3-1	寄存器寻址模式 .....	3-2
图 3-2	工作寄存器寻址模式 .....	3-2
图 3-3	寄存器卷中的间接寄存器寻址模式 .....	3-3
图 3-4	程序存储空间的间接寄存器寻址 .....	3-4
图 3-5	寄存器卷中的间接寄存器寻址 .....	3-5
图 3-6	工作寄存器间接访问程序存储器或数据存储器 .....	3-6
图 3-7	寄存器空间的偏址寻址模式 .....	3-7
图 3-8	偏址寻址模式中短格式访问程序或数据存储空间 .....	3-8

图 3-9	偏址寻址模式中长格式访问程序或数据存储空间.....	3-9
图 3-10	Load 指令的直接寻址.....	3-10
图 3-11	Call, Jump 的直接寻址.....	3-11
图 3-12	间接寻址模式.....	3-12
图 3-13	相对寻址.....	3-13
图 3-14	立即数寻址模式.....	3-14
图 4-1	寄存器描述格式.....	4-4
图 5-1	S3C8系列中的中断类型.....	5-2
图 5-2	S3C828BX/C8289X/C8285X 中断结构.....	5-4
图 5-3	ROM 中断向量地址区.....	5-5
图 5-4	中断功能框图.....	5-8
图 5-5	系统模式寄存器 (SYM).....	5-10
图 5-6	中断屏蔽寄存器(IMR).....	5-11
图 5-7	中断请求优先级组.....	5-12
图 5-8	中断优先级寄存器 (IPR).....	5-13
图 5-9	中断请求寄存器 (IRQ).....	5-14
图 6-1	系统标志寄存器 (FLAGS).....	6-5
图 7-1	晶体/陶瓷振荡器 (fX).....	7-2
图 7-2	外部振荡器 (fX).....	7-2
图 7-3	RC 振荡器 (fX).....	7-2
图 7-4	晶体振荡器 (fxt).....	7-3
图 7-5	外部振荡器 (fxt).....	7-3
图 7-6	系统时钟电路框图.....	7-4
图 7-7	系统时钟控制寄存器 (CLKCON).....	7-5
图 7-8	时钟控制寄存器 (OSCCON).....	7-6
图 7-9	STOP 控制寄存器 (STPCON).....	7-6
图 9-1	P0 口控制寄存器高字节 (P0CONH).....	9-4
图 9-2	P0 口控制寄存器低字节 (P0CONL).....	9-4
图 9-3	P0 口中断控制寄存器高字节 (P0INTH).....	9-5
图 9-4	P0 口中断控制寄存器低字节 (P0INTL).....	9-5
图 9-5	P0 中断标志寄存器 (P0PND).....	9-6
图 9-6	P1 控制寄存器高字节 (P1CONH).....	9-7
图 9-7	P1 口控制寄存器低字节 (P1CONL).....	9-8
图 9-8	P1 上拉电阻使能寄存器 (P1PUR).....	9-8
图 9-9	P2 口控制寄存器高字节 (P2CONH).....	9-9
图 9-10	P2 口控制寄存器低字节 (P2CONL).....	9-10
图 9-11	P3 口控制寄存器高字节 (P3CONH).....	9-11
图 9-12	P3 口控制寄存器低字节 (P3CONL).....	9-12
图 9-13	P4 口控制寄存器高字节 (P4CONH).....	9-13
图 9-14	P4 口控制寄存器低字节 (P4CONL).....	9-14
图 9-15	P4 上拉电阻使能寄存器 (P4PUR).....	9-14
图 9-16	P5 口控制寄存器 (P5CONH).....	9-15
图 9-17	P5 口控制寄存器低字节 (P5CONL).....	9-16
图 9-18	P5 口上拉电阻使能寄存器 (P5PUR).....	9-16

图 9-19	P6 口控制寄存器高字节 (P6CONH).....	9-17
图 9-20	P6 控制寄存器低字节 (P6CONL).....	9-18
图 9-21	P7 口控制寄存器 (P7CON).....	9-19
图 9-22	P8 口控制寄存器 (P8CON).....	9-20
图 10-1	Basic Timer 控制寄存器 (BTCON).....	10-2
图 10-2	Basic Timer 方框图.....	10-4
图 11-1	Timer A 控制寄存器 (TACON).....	11-2
图 11-2	简化的 Timer A 功能框图: Interval (定时) 模式.....	11-3
图 11-3	简化的 Timer A 功能框图: PWM 模式.....	11-4
图 11-4	简化的 Timer A 功能框图: Capture (捕获) 模式.....	11-5
图 11-5	Timer A 功能模块框图.....	11-6
图 11-6	Timer B 控制寄存器.....	11-7
图 11-7	Timer B 功能模块框图.....	11-8
图 11-8	Timer B 在 Repeat (重复) 模式下输出的触发器波形.....	11-10
图 12-1	Timer 0 控制寄存器(T0CON).....	12-2
图 12-2	Timer 0 功能框图.....	12-3
图 12-3	Timer 1 控制寄存器 (T1CON).....	12-5
图 12-4	Timer 1 功能简图: 定时模式.....	12-6
图 12-5	Timer 1 功能简图: PWM 模式.....	12-7
图 12-6	Timer 1 功能简图: 捕获模式.....	12-8
图 12-7	Timer 1 功能框图.....	12-9
图 13-1	钟表定时器控制寄存器(WTCON).....	13-2
图 13-2	钟表定时器电路框图.....	13-3
图 14-1	LCD功能框图.....	14-1
图 14-2	LCD 电路框图.....	14-2
图 14-3	LCD 显示数据 RAM 结构图.....	14-3
图 14-4	LCD 控制寄存器 (LCON).....	14-4
图 14-5	LCD 分压电阻连接.....	14-5
图 14-6	1/2占空比, 1/2偏压比显示模式下选中/非选中信号.....	14-6
图 14-7	1/3占空比, 1/3偏压比显示模式下选中/非选中信号.....	14-7
图 14-8	LCD 信号波形 (1/3 占空比, 1/3 偏压比).....	14-8
图 14-9	LCD 信号波形 (1/4 占空比, 1/3 偏压比).....	14-9
图 14-10	LCD 信号波形 (1/8 占空比, 1/4 偏压比).....	14-10
图 15-1	A/D 转换控制寄存器 (ADCON).....	15-2
图 15-2	A/D 转换数据寄存器 (ADDATAH/L).....	15-3
图 15-3	A/D 转换功能模块图.....	15-4
图 15-4	推荐高精度 A/D 转换电路.....	15-5
图 16-1	SIO 控制寄存器(SIOCON).....	16-2
图 16-2	SIO 预分频寄存器 (SIOPS).....	16-3
图 16-3	SIO 功能模块框图.....	16-3
图 16-4	SIO 串行发送-接收模式 (下降沿发送, SIOCON.4 = 0).....	16-4
图 16-5	SIO 串行发送-接收模式 (上升沿发送, SIOCON.4 = 1).....	16-4



图 17-1	UART 控制寄存器 (UARTCON).....	17-2
图 17-2	UART 中断标志位 (INTPND.5-4).....	17-3
图 17-3	UART 数据寄存器 (UDATA).....	17-4
图 17-4	UART 波特率数据寄存器(BRDATA).....	17-4
图 17-5	UART 功能模块框图.....	17-6
图 17-6	UART Mode 0 模式运行时序图.....	17-8
图 17-7	UART Mode 1 模式运行时序图.....	17-9
图 17-8	UART Mode 2 模式运行时序图.....	17-10
图 17-9	UART Mode 3 模式运行时序图.....	17-11
图 17-10	多节点串行通讯连接示例.....	17-13
图 18-1	电池电压检测模块图.....	18-1
图 18-2	电池电压检测控制寄存器 (BLDCON).....	18-2
图 18-3	电池电压检测电路和模块图.....	18-3
图 19-1	Flash Memory Control Register (FMCON).....	19-2
图 19-2	闪存用户编程使能寄存器 (FMUSR).....	19-3
图 19-3	闪存扇区地址寄存器高字节 (FMSECH).....	19-4
图 19-4	闪存扇区地址寄存器低字节 (FMSECL).....	19-4
图 19-5	程序存储地址空间.....	19-5
图 19-6	用户编程模式下的扇区.....	19-7
图 20-1	外部中断时序.....	20-5
图 20-2	nRESET 输入时序.....	20-5
图 20-3	Stop 模式下复位时序.....	20-6
图 20-4	外部中断唤醒Stop 模式时序.....	20-7
图 20-5	LVR (低电压复位) 时序.....	20-9
图 20-6	串行数据传输时序.....	20-10
图 20-7	UART时序特性波形.....	20-11
图 20-8	UART 模块的时序波形.....	20-12
图 20-9	X <sub>IN</sub> 管脚测得的时钟时序.....	20-14
图 20-10	XT <sub>IN</sub> 管脚测得的时钟时序.....	20-14
图 20-11	工作电压范围.....	20-15
图 21-1	封装尺寸(80-QFP-1420C).....	21-1
图 21-2	封装尺寸(80-TQFP-1212).....	21-2
图 22-1	S3F828BX/F8289X/F8285X 管脚分配 (80-QFP-1420C).....	22-2
图 22-2	S3F828BX/F8289X/F8285X 管脚分配 (80-TQFP-1212).....	22-3
图 23-1	开发系统配置.....	23-2
图 23-2	TB828B/9/5 目标板配置.....	23-3
图 23-3	TB828B/9/5的40脚接口(J101, J102).....	23-7
图 23-4	专为80-QFP 封装的S3E8280的数据线.....	23-7

# List of Tables

Table Number	Title	Page Number
表 1-1	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 管脚特性 .....	1-10
表 2-1	S3C828BX/F828BX 寄存器类型总结 .....	2-5
表 2-2	S3C8289X/F8289X Register 寄存器类型总结 .....	2-6
表 2-3	S3C8285X/F8285X 寄存器类型总结 .....	2-6
表 4-1	Set 1 寄存器 .....	4-1
表 4-2	Set 1, Bank 0 寄存器 .....	4-2
表 4-3	Set 1, Bank 1 寄存器 .....	4-3
表 5-1	中断向量 .....	5-6
表 5-2	中断控制寄存器描述 .....	5-7
表 5-3	中断源控制和数据寄存器 .....	5-9
表 6-1	指令集简介 .....	6-2
表 6-2	标志位符号 .....	6-7
表 6-3	指令集符号 .....	6-7
表 6-4	指令符号的定义 .....	6-8
表 6-5	指令代码快速参考 .....	6-9
表 6-6	指令代码快速参考 (2) .....	6-10
表 6-7	条件码 .....	6-11
表 8-1	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X Set 1 寄存器及其复位后初始值 .....	8-2
表 8-2	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X Set 1, Bank 0 寄存器及其复位后初始值 .....	8-3
表 8-3	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X Set 1, Bank 1 寄存器及其复位后初始值 .....	8-4
表 9-1	S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 口配置描述 .....	9-1
表 9-2	端口数据寄存器 .....	9-2
表 17-1	8-位 BR0DATA可以产生的常用波特率表 .....	17-5
表 18-1	BLDCON 值和检测电压 .....	18-3
表 19-1	ISP扇区大小 .....	19-6
表 19-2	复位向量地址 .....	19-6
表 20-1	芯片极限物理特性 .....	20-2
表 20-2	直流电气特性 .....	20-2
表 20-3	交流电气特性 .....	20-5
表 20-4	输入输出电容 .....	20-6
表 20-5	模式下数据保持电压 .....	20-6
表 20-6	A/D 转换电气特性 .....	20-8

表 20-7	低电压复位电气特性 .....	20-9
表 20-8	电池电压检测电气特性.....	20-9
表 20-9	同步 SIO 电气特性 .....	20-10
表 20-10	UART Mode 0 时序特性(11.1MHz).....	20-11
表 20-11	主振荡器特性 .....	20-13
表 20-12	副振荡器特性 .....	20-13
表 20-13	主振荡稳定时间.....	20-14
表 20-14	副振荡稳定时间.....	20-14
表 20-15	内部 Flash ROM 电气特性 .....	20-15
表 22-1	闪存读写管脚描述.....	22-4
表 22-2	S3F828BX/F8289X/F8285X 和 S3C828BX/C8289X/C8285X特性比较 .....	22-4
表 22-3	编程工作模式选择 .....	22-5
表 23-1	TB828B/9/5 电源选择设定 .....	23-4
表 23-2	TB828B/9/5 主时钟选择控制.....	23-4
表 23-3	TB828B/9/5的芯片选择.....	23-5
表 23-4	SMDS2+ 工具选择设置.....	23-5
表 23-5	TB828B/9/5 Smart Option源设置.....	23-6
表 23-6	TB828B/9/5 的Smart Option设置.....	23-6

# List of Examples

<b>Example Number</b>	<b>Title</b>	<b>Page Number</b>
编程实例 2-1	在清RAM时使用页面指针 (Page 0, Page 1) .....	2-11
编程实例 2-2	设置寄存器指针 .....	2-15
编程实例 2-3	使用 RP 对寄存器 80H-85H 的内容求和 .....	2-16
编程实例 2-4	访问通用工作寄存器区 .....	2-20
编程实例 2-5	用 PUSH 和 POP 实现标准栈操作 .....	2-25
编程实例 5-1	如何清除中断标志位 .....	5-15
编程实例 7-1	CPU时钟切换 .....	7-7
编程实例 11-1	如何在 P3.0 处产生 38kHz, 1/3 占空比信号 .....	11-11
编程实例 11-2	如何在 P3.0 处产生一个脉冲 .....	11-12
编程实例 19-1	扇区擦除 .....	19-8
编程实例 19-2	编程 .....	19-10
编程实例 19-3	读 .....	19-11
编程实例 19-4	Hard Lock 保护 .....	19-12

# 1 产品概述

## 1.1 S3C8-系列 MCU

三星 S3C8 系列单片机向用户提供了高效快速的CPU，丰富的外设接口，以及各种大小的可编程ROM。其中重要的CPU特性包括：

- 高效的寄存器结构
- 可选的CPU时钟源
- 可由中断唤醒的 IDLE 和 STOP 低功耗模式
- 内置有看门狗功能的Basic Timer

精心设计的中断结构支持8个中断优先级，每个中断优先级支持一个或多个中断源及中断向量。任意指定的中断优先级都可以实现快速中断（最少只需4个CPU时钟）。

## 1.2 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X MCU

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X单片机采用了三星最新的 CPU 构架和先进的 CMOS 工艺。

S3C828BX, S3C8289X, S3C8285X 采用了 64K字节, 32K字节, 16K字节 可编程mask ROM 的微控制器

S3F828BX 是一款内置64K 字节 Flash ROM 的微控制器。

S3F8289X 是一款内置32K 字节 Flash ROM 的微控制器。

S3F8285X 是一款内置16K 字节 Flash ROM 的微控制器。

通过成熟的模块化设计方法, 和基于强大的SAM8 内核, 在S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 中成功地集成了以下外围模块:

- 9个可编程 I/O口, 包括6个8位 I/O 口, 1个7位 I/O 口, 1个6位 I/O 口和1个4位 I/O 口, 共65个管脚。
- 8个供外部中断使用的位可编程管脚。
- 1个可用于振荡稳定和看门狗 (系统复位) 的8位 basic timer。
- 2个8位timer/counter 和2个可选工作模式的16位 timer/counter。
- 可用作实时时钟的钟表定时器 (Watch timer)
- LCD 控制器/驱动器
- 8通道模数转换通道
- 同步 SIO 模块

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 适用于多种应用, 包括照相机、LCD 和ADC应用等。现有的封装类型有 80-pin TQFP 和 80-pin QFP。

### 1.2.1 FLASH

S3F828BX/F8289X/F8285X 是S3C828BX/C8289X/C8285X 的 FLASH 版本。S3F828BX/F8289X/F8285X 具有片上 FLASH ROM 而不是 mask ROM, S3F828BX/F8289X/F8285X与S3C828BX/C8289X/C8285X 在所有的功能和管脚分配上都是兼容的, 其中 S3F828BX为 full flash 产品, Full flash 产品在程序中可以使用指令对 ROM 写入数据。

## 1.2.2 特性

### 1.2.2.1 CPU

- SAM88 RC CPU 内核

### 1.2.2.2 存储器

- 程序存储器(ROM)
  - 64K × 8位程序存储器 (S3C828BX/F828BX)
  - 32K × 8位程序存储器 (S3C8289X/F8289X)
  - 16K × 8位程序存储器 (S3C8285X/F8285X)
  - 内部flash 存储器 (程序存储器)
    - o 扇区(Sector): 128 bytes
    - o 10 年数据保持
    - o 快速编程:
      - + 整片擦除: 50ms
      - + 扇区擦除: 10ms
      - + 字节编程: 30μs
    - o 用户通过 'LDC'指令编程
    - o 寿命: 10,000 次可擦除/编程
    - o 扇区 (128 bytes) 擦除
    - o 字节编程
    - o 支持外部串行编程
    - o 可扩展的 OBPTM (on board program) 扇区
- 数据存储器 (RAM)
  - 包括 LCD 显示数据存储器
  - 2614 × 8位 数据存储器 (S3C828BX/F828BX)
  - 1078 × 8位 数据存储器 (S3C8289X/F8289X)
  - 566 × 8位 数据存储器 (S3C8285X/F8285X)

### 1.2.2.3 指令及

- 78 条指令
- 用于低功耗模式的Idle 和Stop 指令

### 1.2.2.4 65 I/O 口

- I/O: 25 个
- I/O: 40 个 (与LCD信号复用)

#### 1.2.2.5 中断

- 8 个中断级，12个中断源
- 支持快速中断处理

#### 1.2.2.6 8 位 Basic Timer

- 看门狗功能
- 4 种可选时钟源

#### 1.2.2.7 8 位 Timer/Counter A

- 可编程 8位定时 Timer
- 外部事件计数功能
- PWM 和 事件捕获功能

#### 1.2.2.8 8 位 Timer/Counter B

- 可编程 8位定时 Timer
- 载波发生器

#### 1.2.2.9 16 位 Timer/Counter 0

- 可编程 16位定时 Timer

#### 1.2.2.10 16 位 Timer/Counter 1

- 可编程 16位定时 Timer
- 外部事件计数功能
- PWM 和 事件捕获功能

#### 1.2.2.11 钟表定时器 (Watch Timer)

- 时间间隔: 3.91mS, 0.25S, 0.5S, 和 1S (32.768kHz)
- 0.5/1/2/4kHz 可选的蜂鸣器输出

#### 1.2.2.12 LCD 控制/驱动器

- 32 个段，8 个公共端
- 1/2, 1/3, 1/4, 和1/8 占空比可选
- 内置LCD偏置需要的电阻分压电路



### 1.2.2.13 A/D 转换

- 8通道模拟信号输入
- 10位转换精度
- 25uS 转换时间

### 1.2.2.14 UART

- 全双工串行通讯接口
- 四种可编程运行模式

### 1.2.2.15 8位串行输入输出接口

- 8位发送/接收模式
- 8位接收模式
- LSB 优先或MSB 优先发送可选
- 内部或外部时钟源

### 1.2.2.16 电池电压检测

- 3种可选电压阈值 (2.2V, 2.4V, 2.8V)
- 根据功耗要求, 软件使能或禁止

### 1.2.2.17 低电压复位 (LVR)

- Criteria voltage电压阈值: 2.2V
- 通过Smart Option (ROM 地址: 3FH)使能/禁止

### 1.2.2.18 2种低功耗模式

- Idle: 仅停止 CPU 时钟
- Stop: 停止被选中的系统时钟和CPU时钟

### 1.2.2.19 振荡源

- 主时钟可选石英晶振, 陶瓷振荡器或者RC
- 主时钟频率: 0.4MHz – 11.1MHz
- 副时钟选用32.768kHz 晶体振荡电路

### 1.2.2.20 指令执行时间

- $f_x = 11.1\text{MHz}$  时, 最快360nS

**1.2.2.21 工作电压范围**

- 2.0V - 3.6V ( 0.4 – 4.2MHz)
- 2.7V - 3.6V ( 0.4 – 10MHz)
- 3.0V - 3.6V ( 0.4 – 11.1MHz)

**1.2.2.22 工作温度范围**

- -25°C - +85°C

**1.2.2.23 封装形式**

- 80-QFP-1420C, 80-TQFP-1212

**1.2.2.24 Smart Option**

- 低电压复位 (LVR) 的使能/禁止及复位电压可由硬件选择 (ROM 地址 3FH)
- ISP 相关设置选择 (ROM 地址 3EH)

### 1.3 内部模块框图

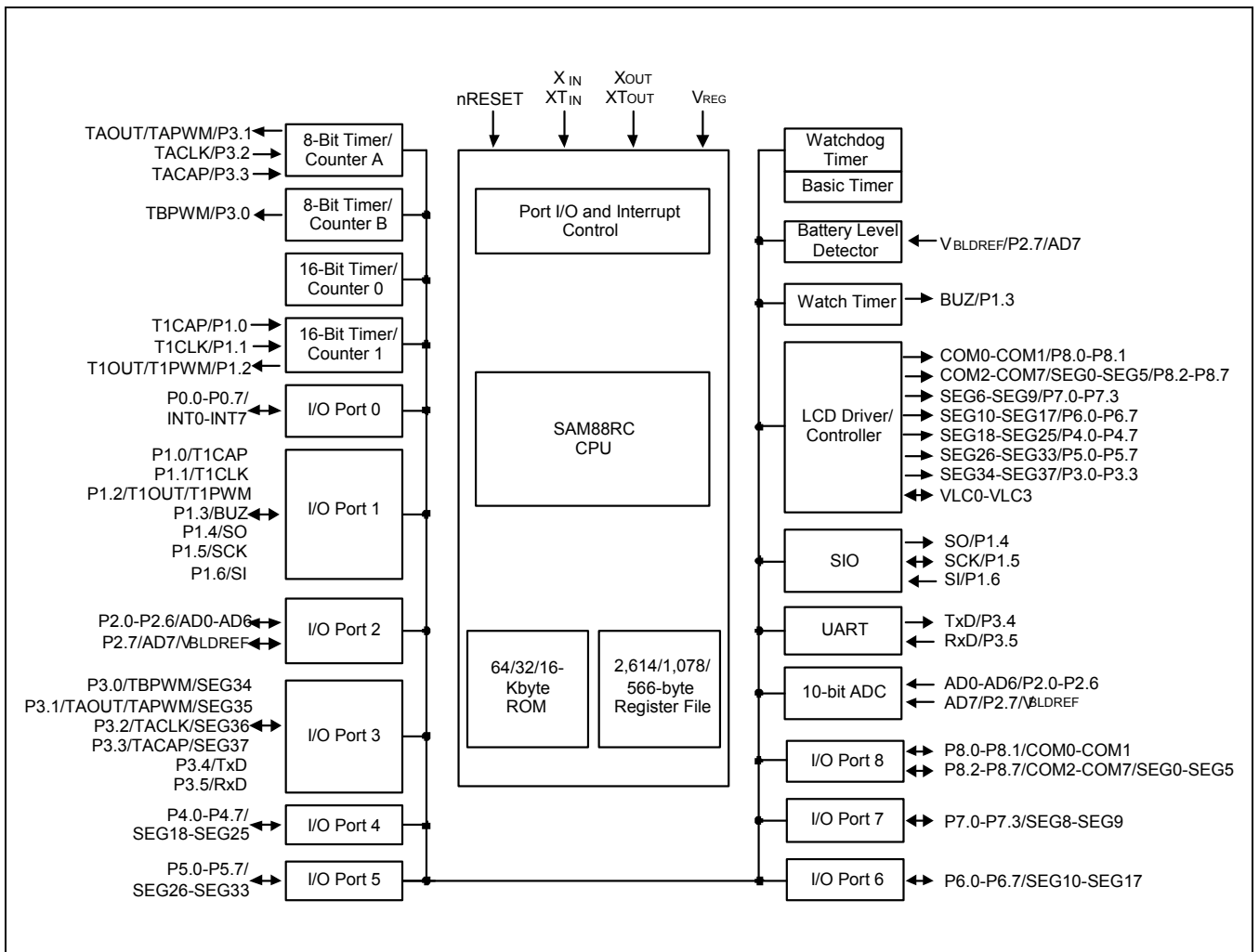


图 1-1 内部模块框图

## 1.4 管脚分布图

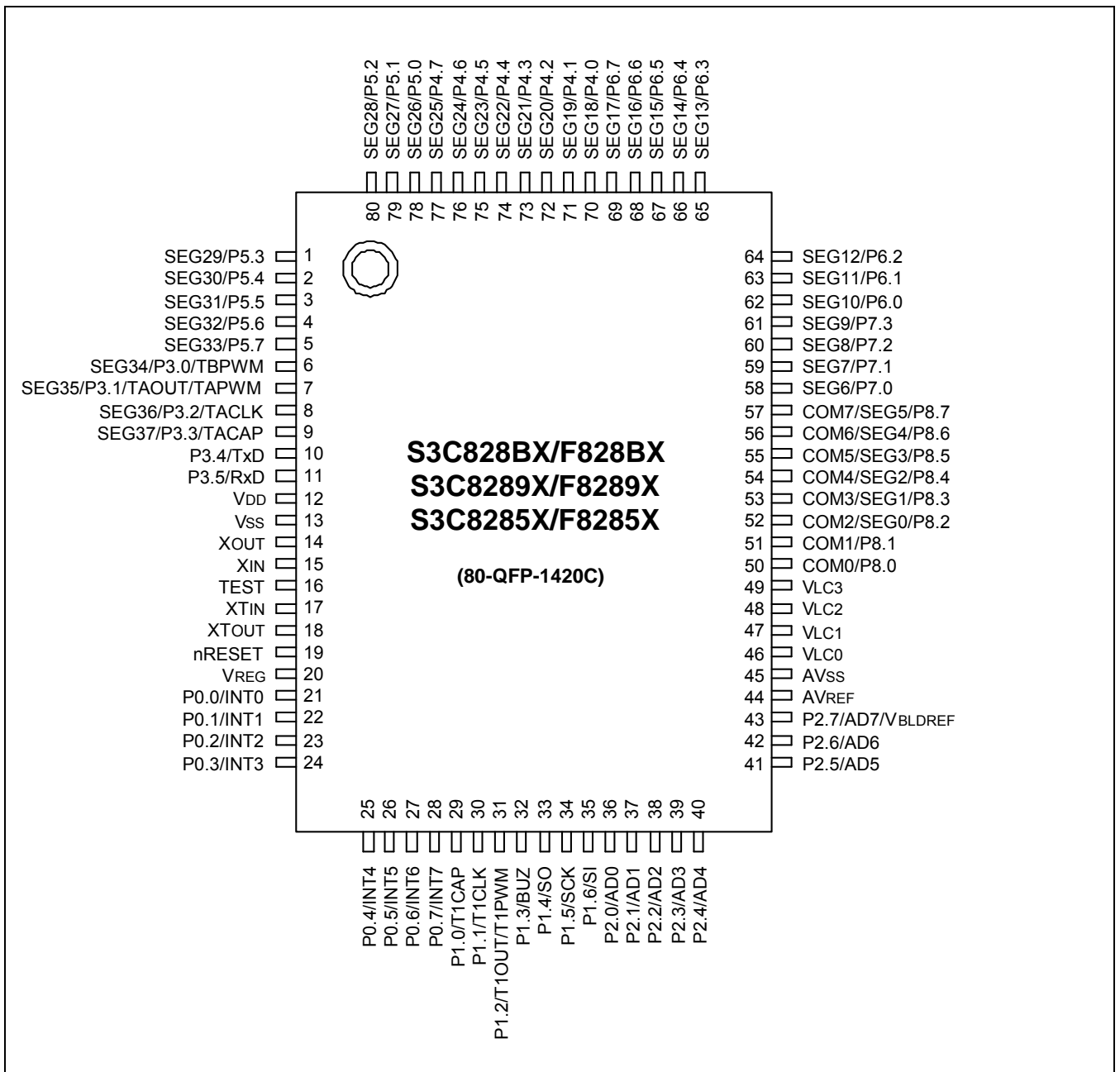


图 1-2 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 管脚分布 (80-QFP-1420C)

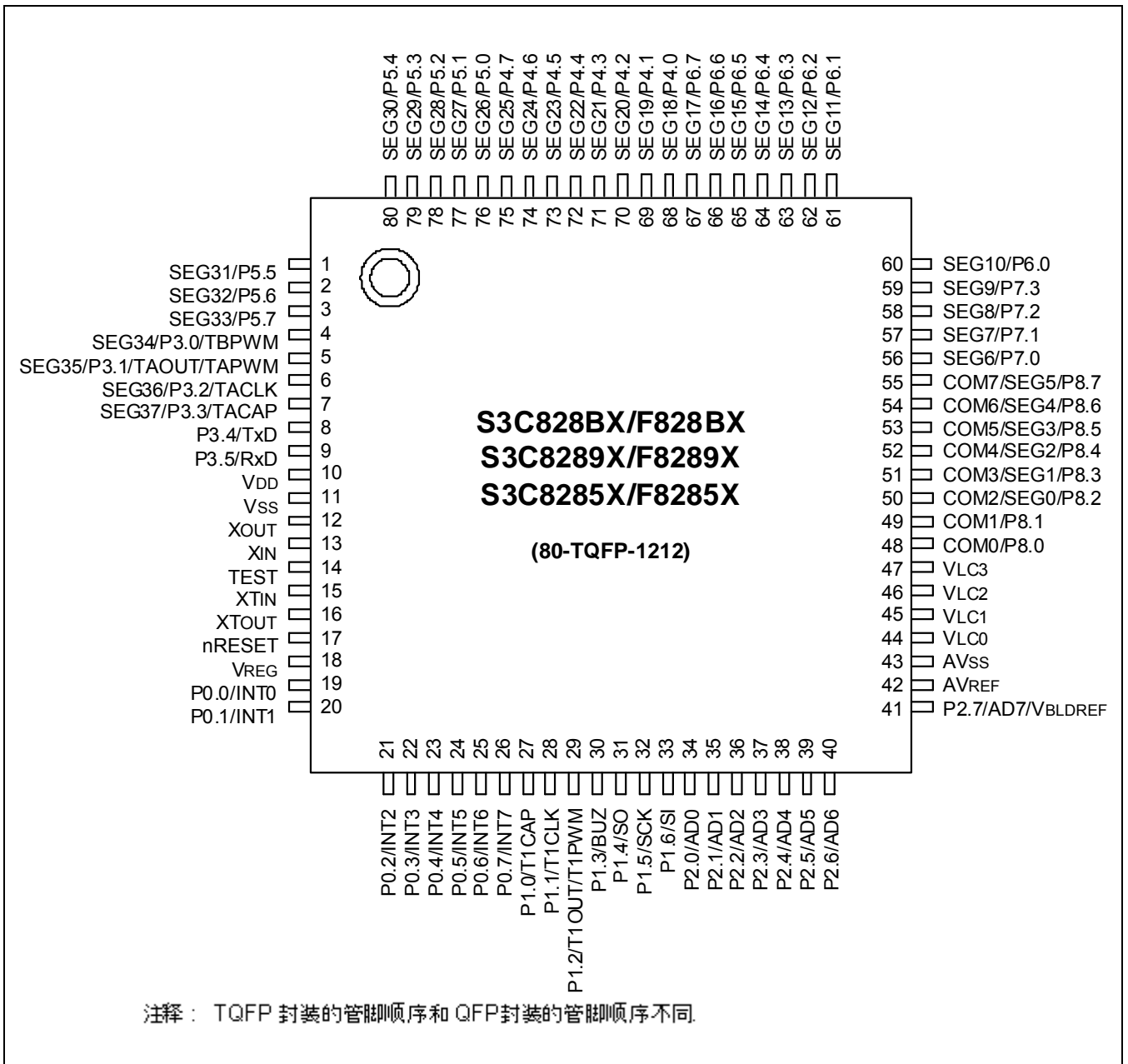


图 1-3 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X管脚分布(80-TQFP-1212)

## 1.5 管脚特性描述

表 1-1 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 管脚特性

管脚名称	输入/输出	管脚特性描述	管脚类型	管脚号	复用功能
P0.0–P0.7	I/O	可对该口的每一位进行功能设定，可以设定为施密特触发器输入、推挽式输出或开漏输出。通过软件设定上拉电阻。 P0.0 - P0.7 也可用作外部中断输入(噪声滤波器，中断使能和标志位控制)。	E-4	21–28 (19–26)	INT0–INT7
P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6	I/O	可对该口的每一位进行功能设定，可以设定为施密特触发器输入、推挽式输出或开漏输出。通过软件设定上拉电阻。	E-4	29(27) 30(28) 31(29) 32(30) 33(31) 34(32) 35(33)	T1CAP T1CLK T1OUT/T1PWM BUZ SO SCK SI
P2.0–P2.6 P2.7	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	F-1 F-2	36–42 (34–40) 43(41)	AD0–AD6 AD7/VBLDREF
P3.0 P3.1 P3.2 P3.3 P3.4 P3.5	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9 D-1	6(4) 7(5) 8(6) 9(7) 10(8) 11(9)	TBPWM/SEG34 TAOUT/TAPWM/SEG 35 TACLK/SEG36 TACAP/SEG37 TxD RxD
P4.0–P4.7	I/O	可对该口的每一位进行功能设定，可以设定为输入、推挽式输出或开漏输出。通过软件设定上拉电阻。	H-8	70–77 (68–75)	SEG18–SEG25
P5.0–P5.7	I/O	可对该口的每一位进行功能设定，可以设定为输入、推挽式输出或开漏输出。通过软件设定上拉电阻。	H-8	78–80,1–5 (76–80,1–3)	SEG26–SEG33
P6.0–P6.7	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9	62–69 (60–67)	SEG10–SEG17
P7.0–P7.3	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9	58–61 (56–59)	SEG6–SEG9
P8.0–P8.1 P8.2–P8.7	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9	50–51(48–49) 52–57(50–55)	COM0–COM1 COM2–COM7/ SEG0–SEG5
V <sub>LC0</sub> –V <sub>LC3</sub>	–	LCD 电源。	–	46–49(44–47)	–
INT0–INT7	I/O	外部中断输入管脚	E-4	21–28(19–26)	P0.0–P0.7

管脚名称	输入/输出	管脚特性描述	管脚类型	管脚号	复用功能
T1CAP	I/O	Timer 1 捕获输入.	E-4	29(27)	P1.0
T1CLK	I/O	Timer 1 外部时钟输入	E-4	30(28)	P1.1
T1OUT/T1PWM	I/O	Timer 1 时钟输出和PWM输出	E-4	31(29)	P1.2
BUZ	I/O	蜂鸣器输出管脚	E-4	32(30)	P1.3
SO, SCK, SI	I/O	串行时钟, 串行数据输出, 串行数据输入	E-4	33-35 (31-33)	P1.4, P1.5, P1.6
AD0-AD6	I/O	A/D 转换模拟信号输入通道	F-1	36-42 (34-40)	P2.0-P2.6
AD7	I/O		F-2	43(41)	P2.7/VBLDREF
AVREF	-	A/D 转换参考电压	-	44(42)	-
AVSS	-	A/D 转换参考地	-	45(43)	-
VBLDREF	-	电池电压检测参考电压	F-2	43(41)	P2.7/AD7
TACAP	I/O	Timer A 捕获输入管脚	D-1	9(7)	P3.3
TACLK	I/O	Timer A 外部时钟输入	D-1	8(6)	P3.2
TAOUT/TAPWM	I/O	Timer A 时钟输出和PWM输出	D-1	7(5)	P3.1
TBPWM	I/O	Timer B PWM输出.	D-1	6(4)	P3.0
TxD, RxD	I/O	Uart 数据输出, 输入	D-1	10,11(8,9)	P3.4, P3.5
COM0-COM1 COM2-COM7	I/O	LCD 公共端信号输出c	H-9	50-51(48-49) 52-57(50-55)	P8.0-P8.1 P8.2-P8.7/ SEG0-SEG5
SEG0-SEG5	I/O	LCD 段信号输出.	H-9	52-57(50-55)	COM2-COM7/ P8.2-P8.7
SEG6-SEG9				58-61(56-59)	P7.0-P7.3
SEG10-SEG17				62-69(60-67)	P6.0-P6.7
SEG18-SEG25			H-8	70-77(68-75)	P4.0-P4.7
SEG26-SEG33				78-80,1-5 (76-80,1-3)	P5.0-P5.7
SEG34			H-9	6(4)	P3.0/TBPWM
SEG35				7(5)	P3.1/TAOUT/TAPWM
SEG36				8(6)	P3.2/TACLK
SEG37				9(7)	P3.3/TACAP
V <sub>REG</sub>	O	使用副时钟时, 稳压器电压输出 (需接0.1μF电容)	-	20(18)	-
nRESET	I	系统复位脚	B	19(17)	-
XT <sub>IN</sub> , XT <sub>OUT</sub>	-	副时钟晶振管脚	-	17,18(15,16)	-
X <sub>IN</sub> , X <sub>OUT</sub>	-	主振荡器管脚	-	15,14(13,12)	-
TEST	I	测试输入, 必须接到VSS	-	16(14)	-
V <sub>DD</sub> , V <sub>SS</sub>	-	电源输入管脚. VDD 和 VSS之间必须接一个电容	-	12,13(10,11)	-

注释: 括号内的数字为 80-TQFP-1212 封装对应的管脚号。

1.6 管脚电路

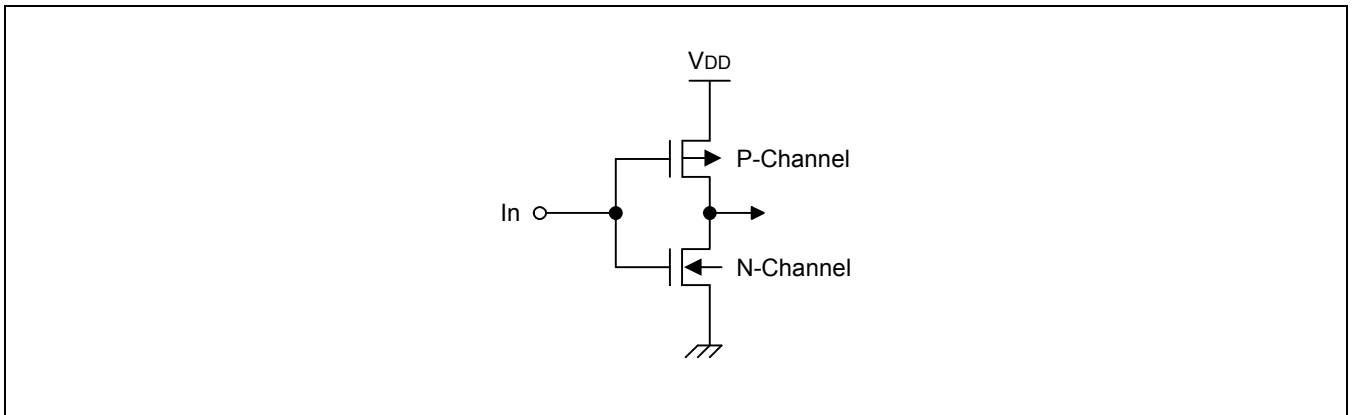


图 1-4 管脚电路类型 A

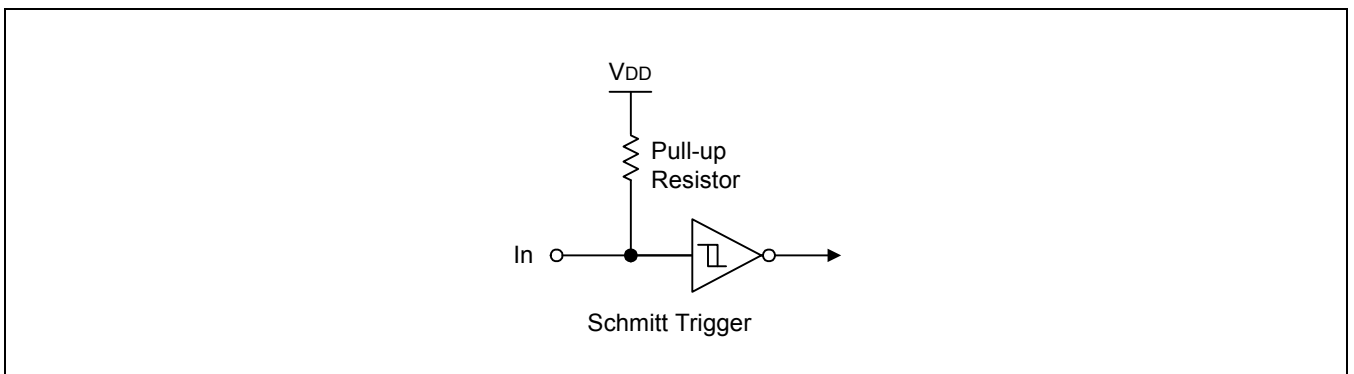


图 1-5 管脚电路类型B

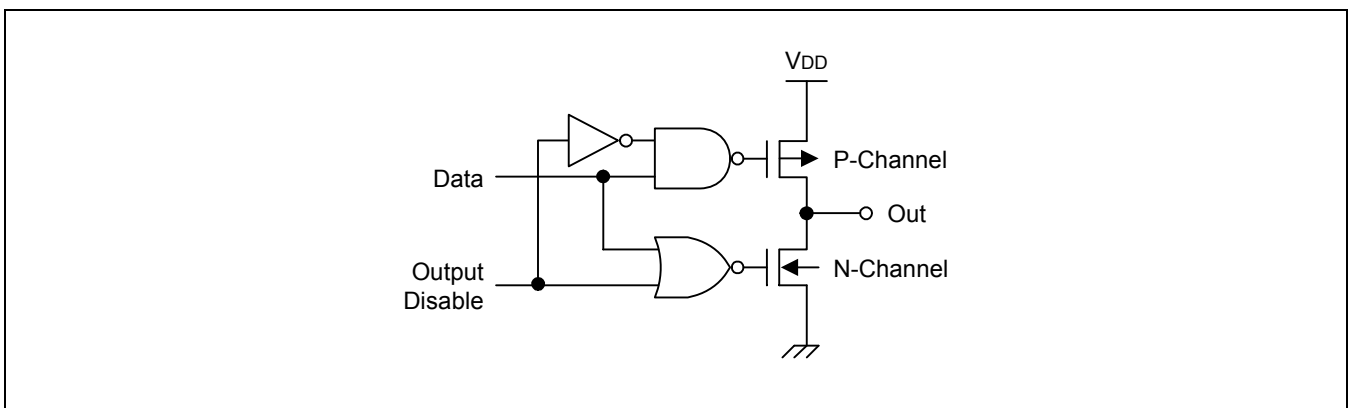


图 1-6 管脚电路类型C



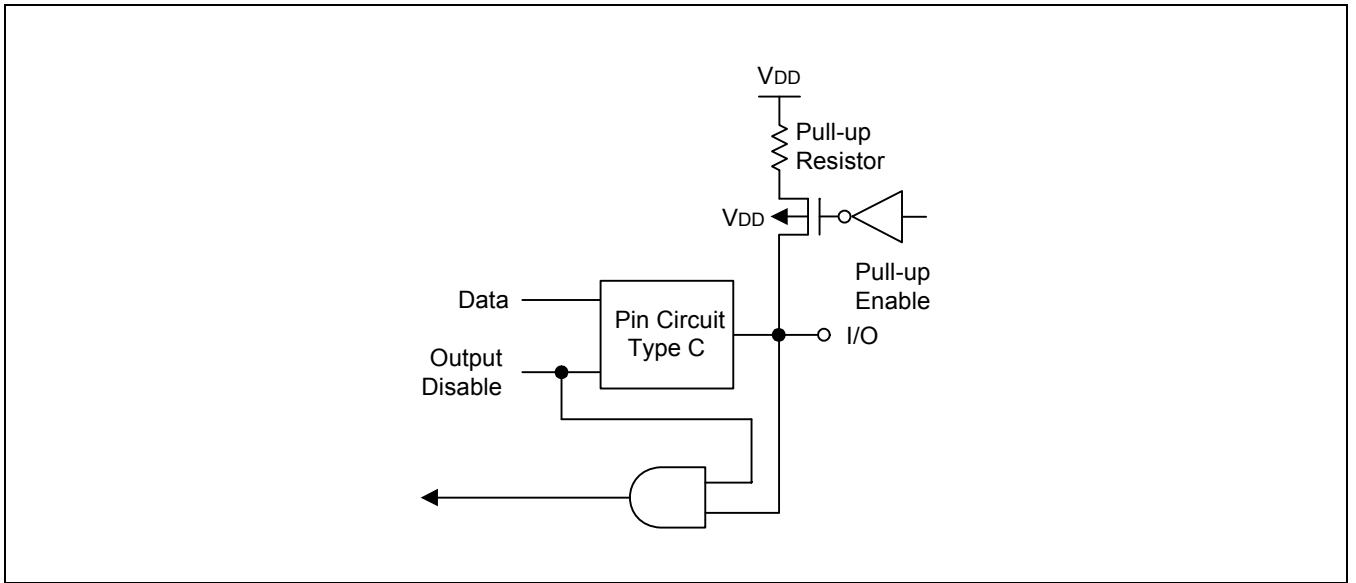


图 1-7 管脚电路类型D-1 (P3.4, P3.5)

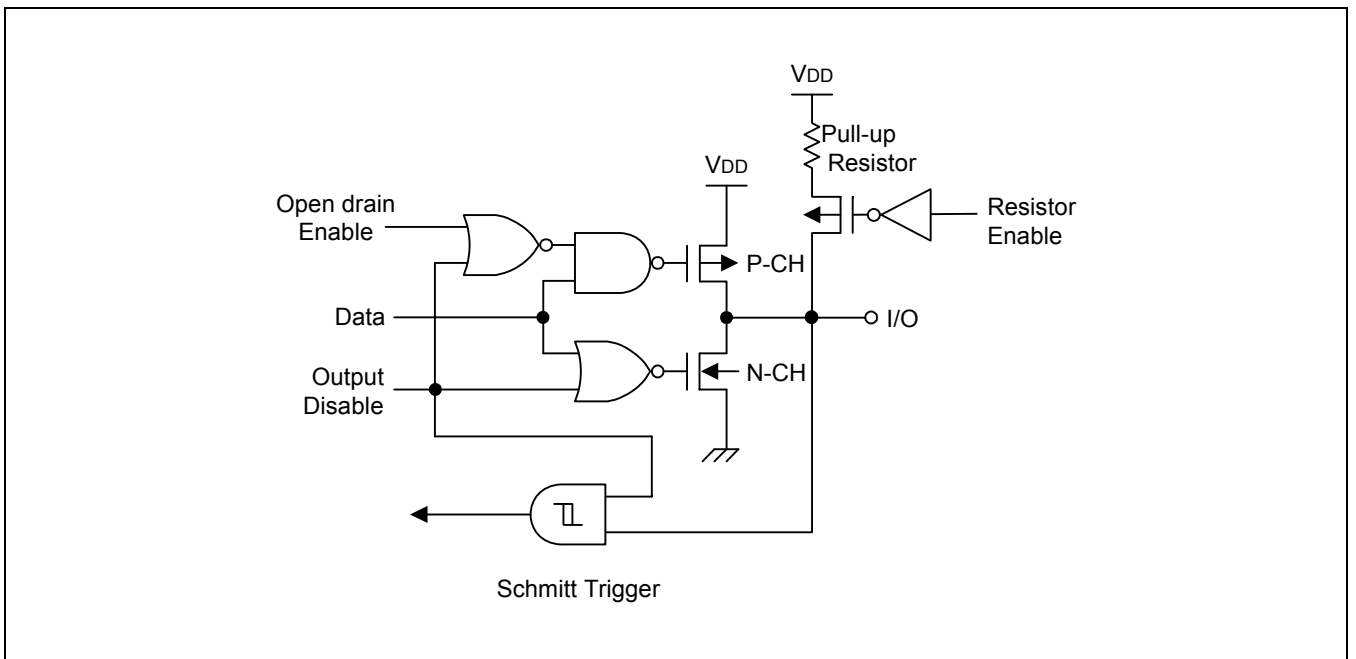


图 1-8 管脚电路类型E-4 (P0, P1)

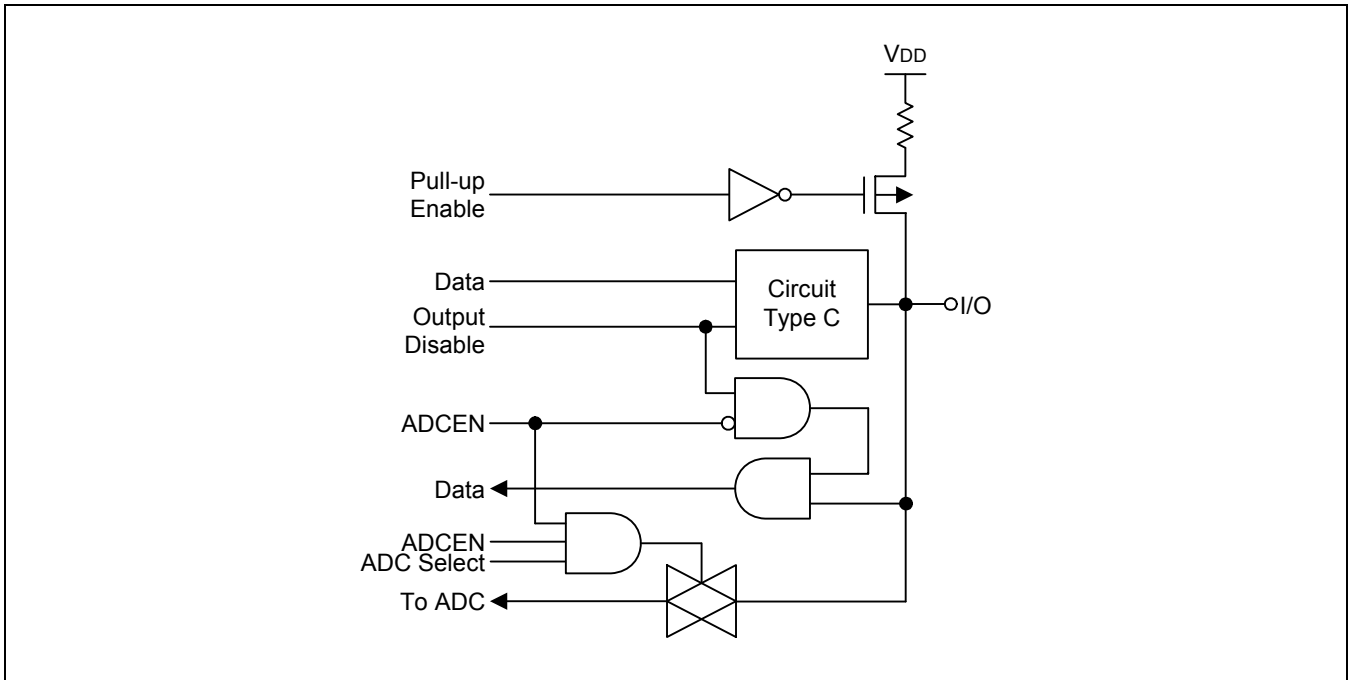


图 1-9 管脚电路类型F-1 (P2.0-P2.6)

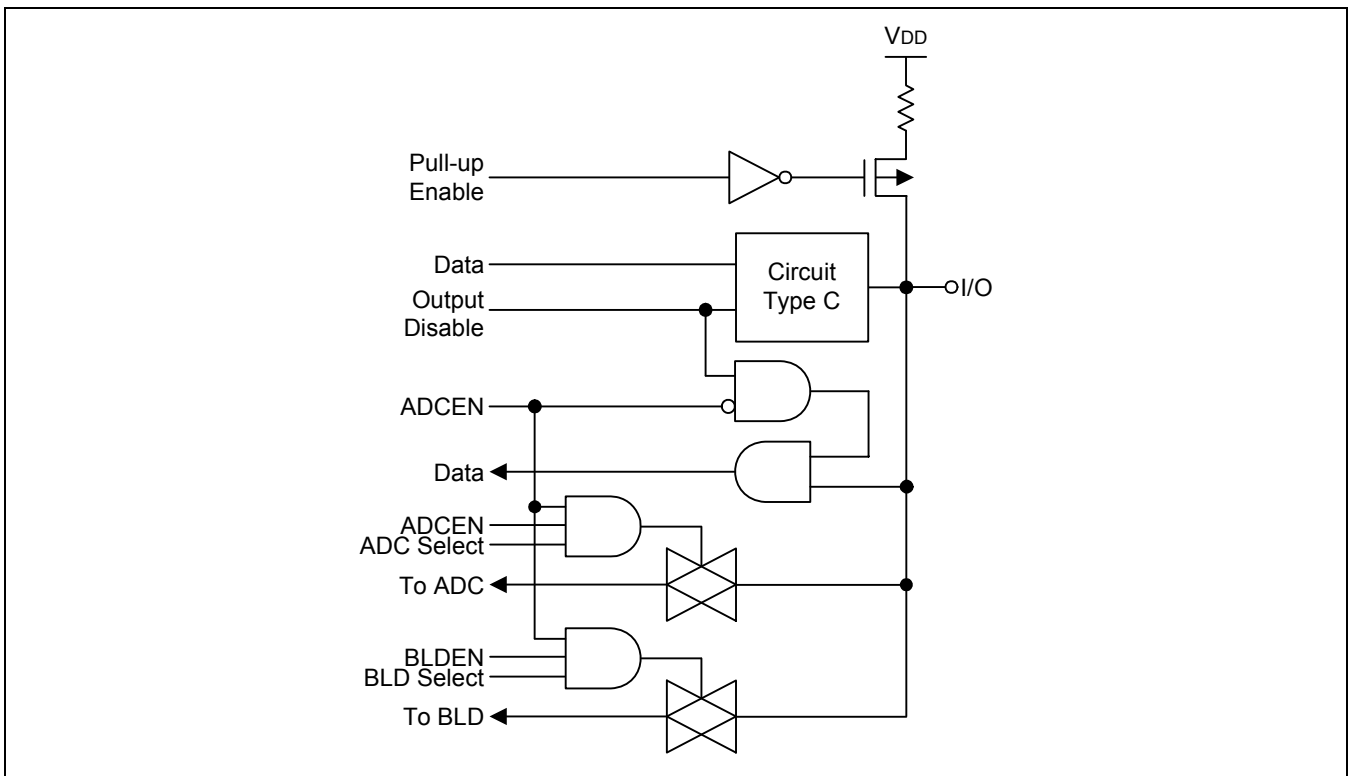


图 1-10 管脚电路类型F-2 (P2.7)

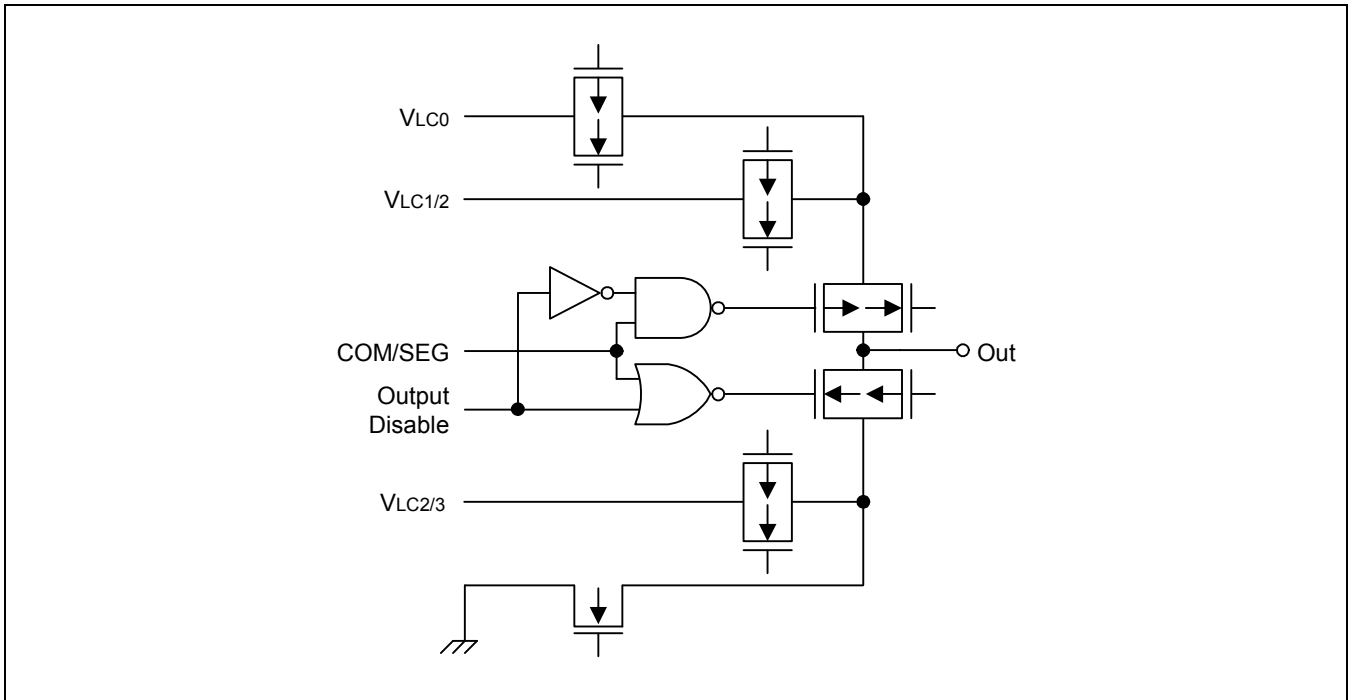


图 1-11 管脚电路类型H-4

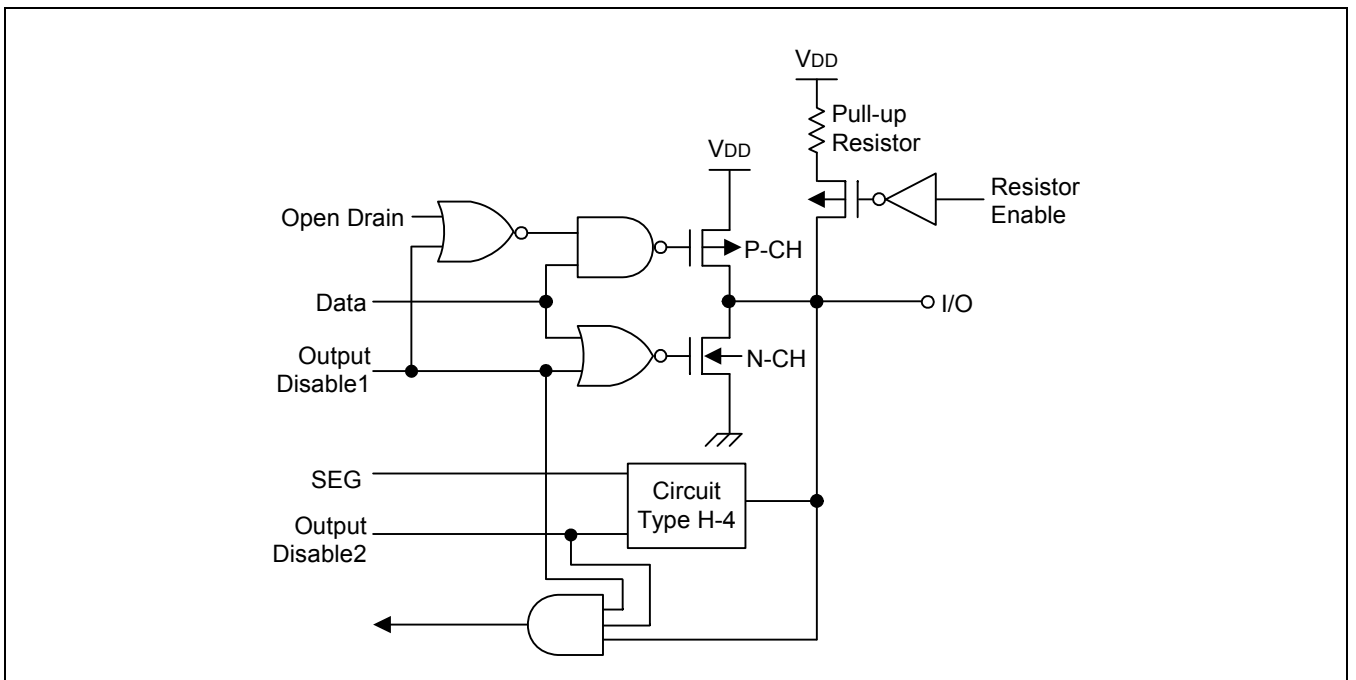


图 1-12 管脚电路类型H-8 (P4, P5)

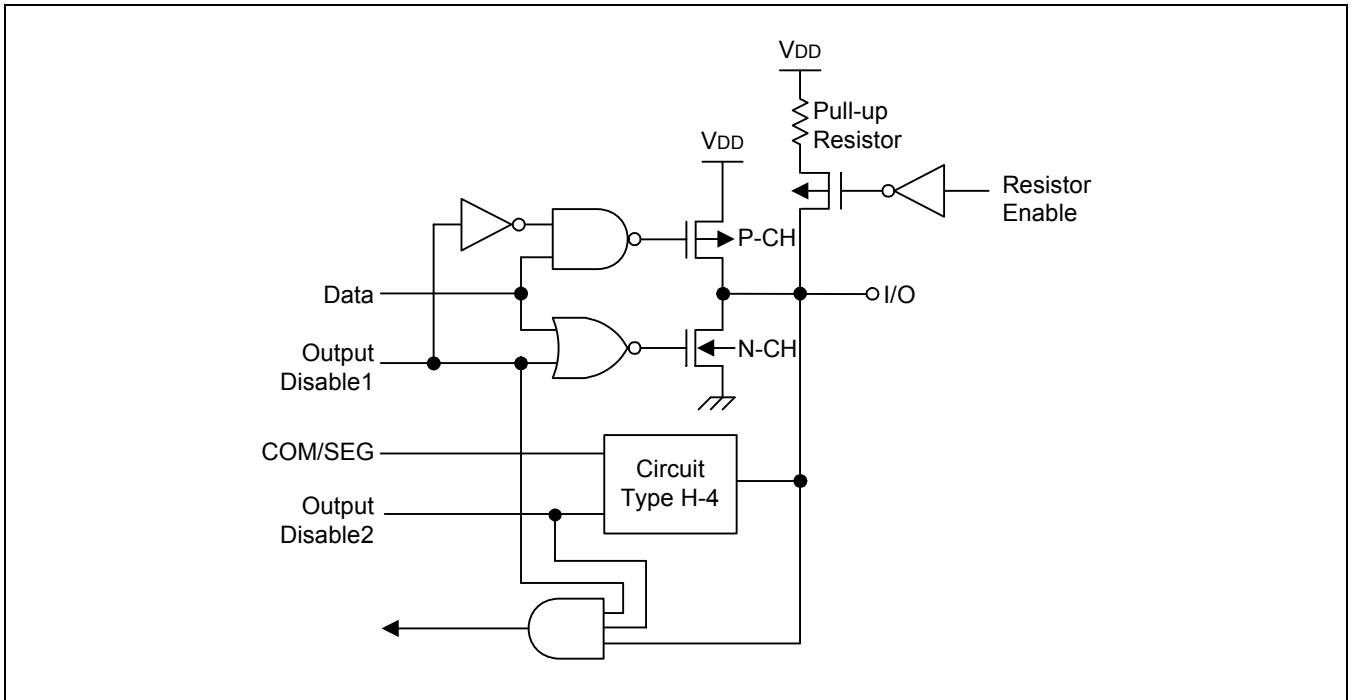


图 1-13 管脚电路类型H-9 (P3.0-P3.3, P6, P7, P8)

# 2 地址空间

## 2.1 概述

S3C828BX/C8289X/C8285X MCU有两类地址空间:

- 内部 程序存储空间 (ROM)
- 内部寄存器卷

MCU 通过16位的地址总线访问程序存储空间, 通过独立的8位地址线和数据线访问内部寄存器卷。

S3C828BX内集成了64K字节的掩模可编程(mask-programmable) 存储器。 S3C8289X内集成了32K字节的掩模可编程(mask-programmable) 存储器。 S3C8285X内集成了16K字节的掩模可编程(mask-programmable) 存储器。

256字节的物理寄存器空间通过不同的寻址方式被扩展成可寻址的320字节的空間。

内建一个38字节的 LCD 显示寄存器卷。

## 2.2 程序存储空间 (ROM)

程序存储空间 (ROM) 保存程序代码或表格数据。S3C828BX/F828BX 有64K字节的内部掩模可编程程序存储空间，S3C8289X/F8289X 有32K字节的内部掩模可编程程序存储空间，S3C8285X/F8285X有16K字节的内部掩模可编程程序存储空间。

ROM (0H–0FFH)中初始 256 字节预留作中断入口地址。未使用的地址都可用作普通的程序存储空间。在使用中断地址区域作为程序代码空间时，注意不可覆盖中断入口地址。

在 ROM 中程序的复位地址是 0100H。

在S3F828BX (Full-Flash 产品)中，ROM复位后开始运行的起始地址可以通过smart option进行修改。详细信息请参考第19章：嵌入式闪存接口

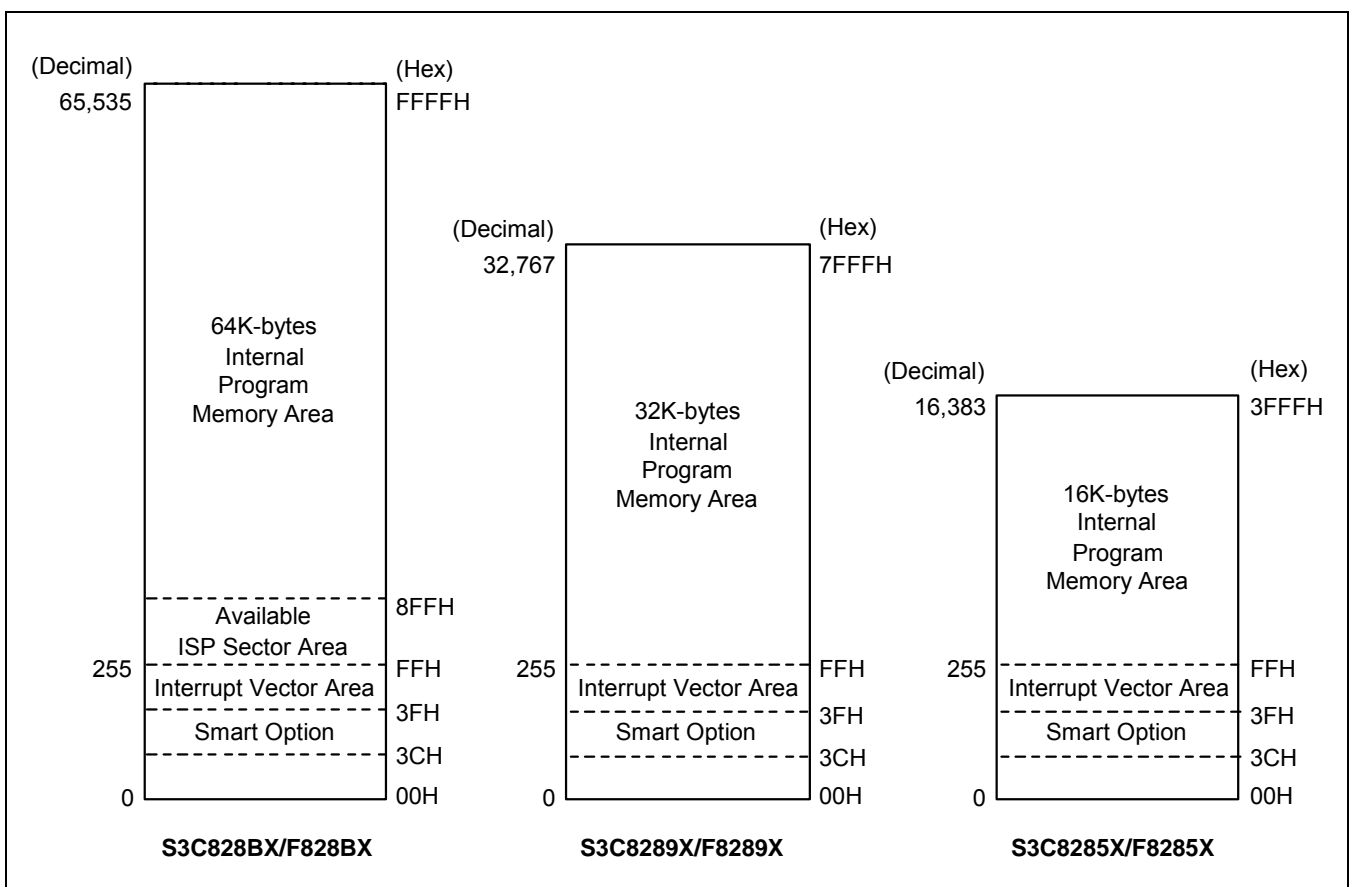


图 2-1 程序存储地址空间

## 2.3 SMART OPTION

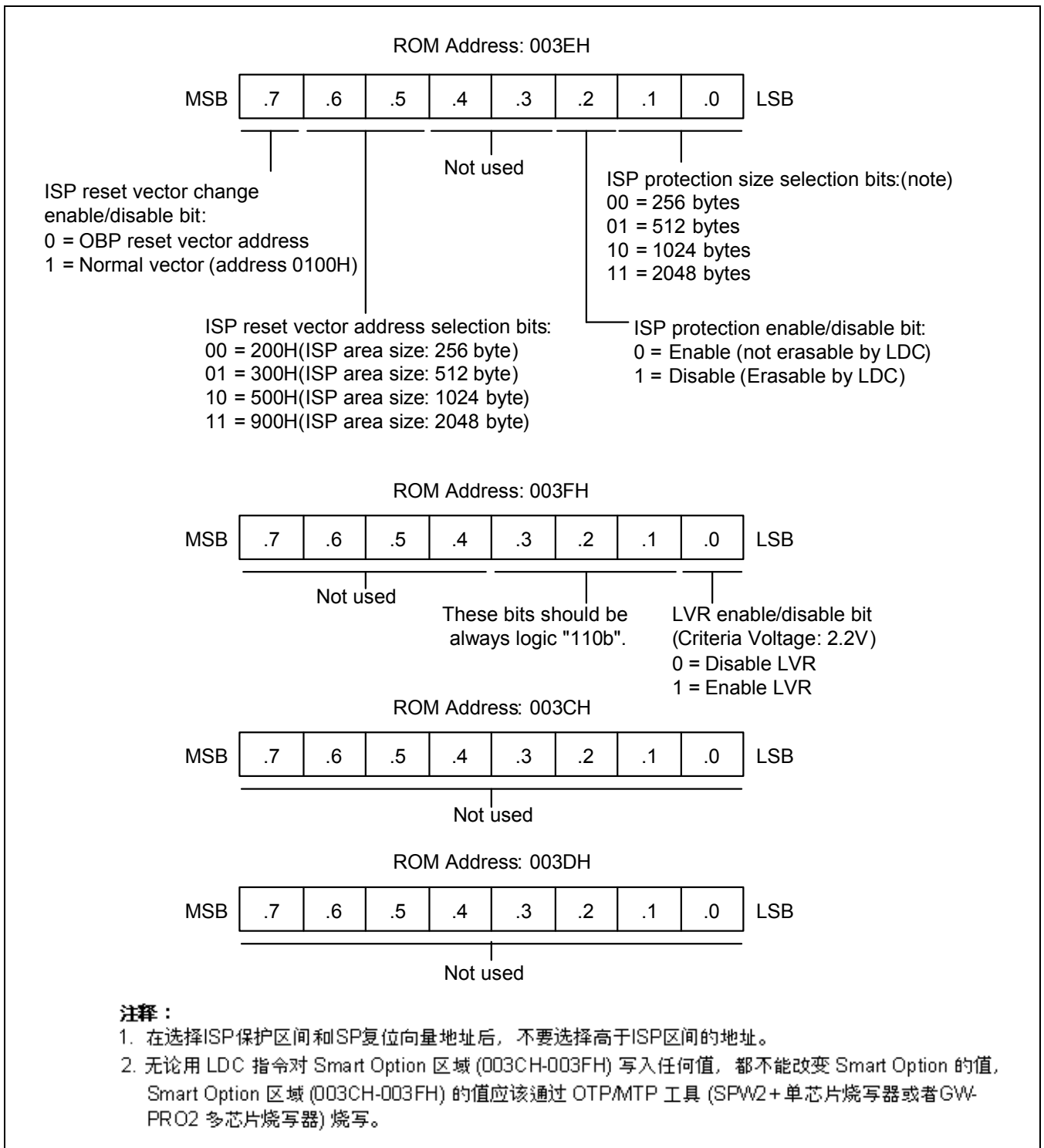


图 2-2 Smart Option

Smart Option 是用来初始化芯片的ROM选项，决定了芯片的启动状态。ROM 中用于 Smart Option 的地址范围为 003CH ~ 003FH。

Smart option(003EH)的ISP功能只有在S3F828BX有效。ROM的003EH地址的默认值为FFH。  
在使用S3C828BX/C8289X/F8289X/C8285X/F8285X时，ROM的003EH地址的值应始终保持为FFH。

Smart option(003FH)LVR 的选择功能对所有器件都有效。  
S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X，ROM的003FH地址的默认值为FFH



## 2.4 寄存器结构

在 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 中，寄存器卷的高 64 字节被扩展为两个 64 字节区域，叫做 set 1 和 set 2。Set 1 的高 32 字节区域又进一步扩展为两个 32 字节寄存器块(bank 0 和 bank 1)，低 32 字节是独立的公共存储区域。

S3C828BX/F828BX可寻址的8位寄存器个数为2,695。其中包含，13字节的CPU和系统控制寄存器，38字节的LCD 数据寄存器，68字节的外设控制和数据寄存器，16字节的共享工作寄存器以及page 0-page 9中的2,560个通用寄存器 (在S3C8289X/F8289X中只有page 0-page 3, 在S3C8285X/F8285X中只有page0-page1)。

不管当前选的是哪个寄存器页，都可以对 set 1 的寄存器进行寻址。但只能通过寄存器寻址模式进行。

通过不同寻址模式的限制，bank 选择指令(SB0, SB1)，以及寄存器页面指针(PP)，得以将寄存器空间扩展为各个独立的可寻址区域 (set, bank 和 page)。

[表 2-1](#) 中总结了内部寄存器卷中特殊功能寄存器类型和所占字节数。

**表 2-1 S3C828BX/F828BX 寄存器类型总结**

寄存器类型	所占字节数
通用寄存器 (包括 16字节的通用工作寄存器卷，10个192字节的主寄存器卷 (包括外设控制寄存器)，以及10个64字节的 set 2 卷)	2,576
LCD 数据寄存器	38
CPU 和系统控制寄存器	13
时钟、外设、I/O 控制和数据映射寄存器	68
所有可寻址的字节数	2,695

表 2-2 S3C8289X/F8289X Register 寄存器类型总结

寄存器类型	所占字节数
通用寄存器 (包括 16字节的通用工作寄存器卷, 4个192字节的主寄存器卷 (包括外设控制寄存器), 以及4个64字节的 set 2 卷)	1,040
LCD 数据寄存器	38
CPU 和系统控制寄存器	13
时钟、外设、I/O 控制和数据映射寄存器	68
所有可寻址的字节数	1,159

表 2-3 S3C8285X/F8285X 寄存器类型总结

寄存器类型	所占字节数
通用寄存器 (包括 16字节的通用工作寄存器卷, 2个192字节的主寄存器卷 (包括外设控制寄存器), 以及2个64字节的 set 2 卷)	528
LCD 数据寄存器	38
CPU 和系统控制寄存器	13
时钟、外设、I/O 控制和数据映射寄存器	68
所有可寻址的字节数	647

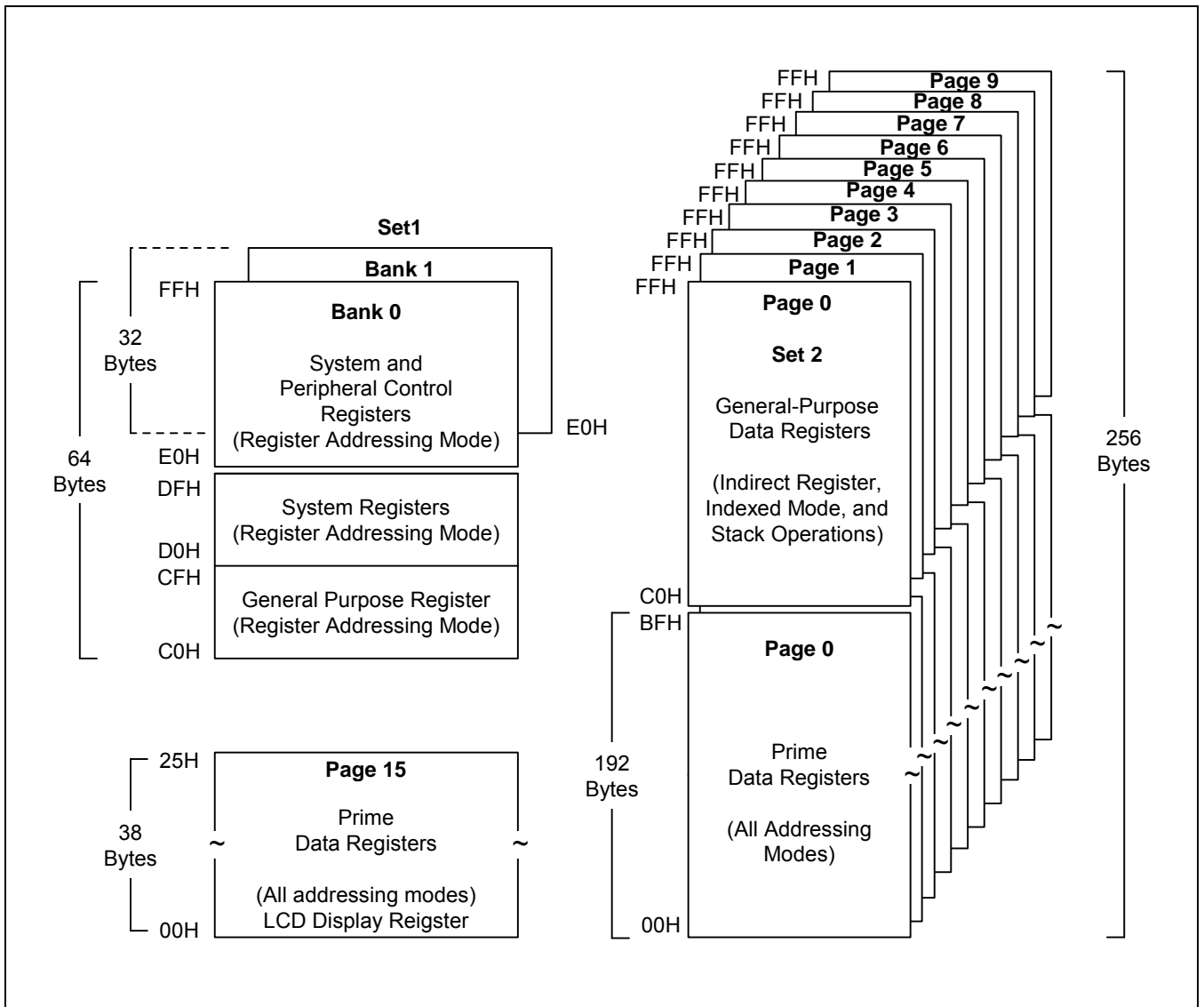


图 2-3 内部寄存器卷的地址空间 (S3C828BX/F828BX)

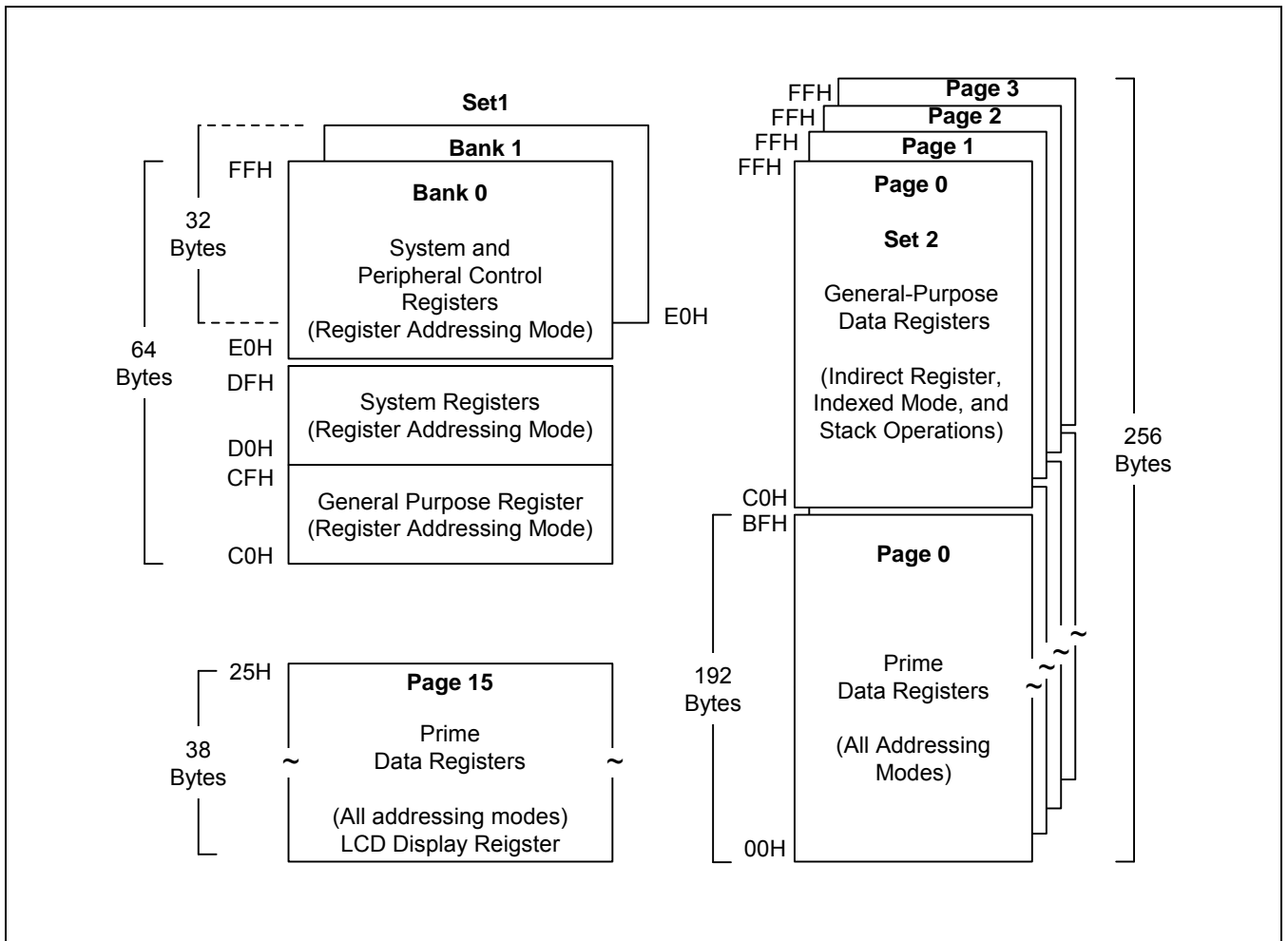


图 2-4 内部寄存器卷的地址空间(S3C8289X/F8289X)

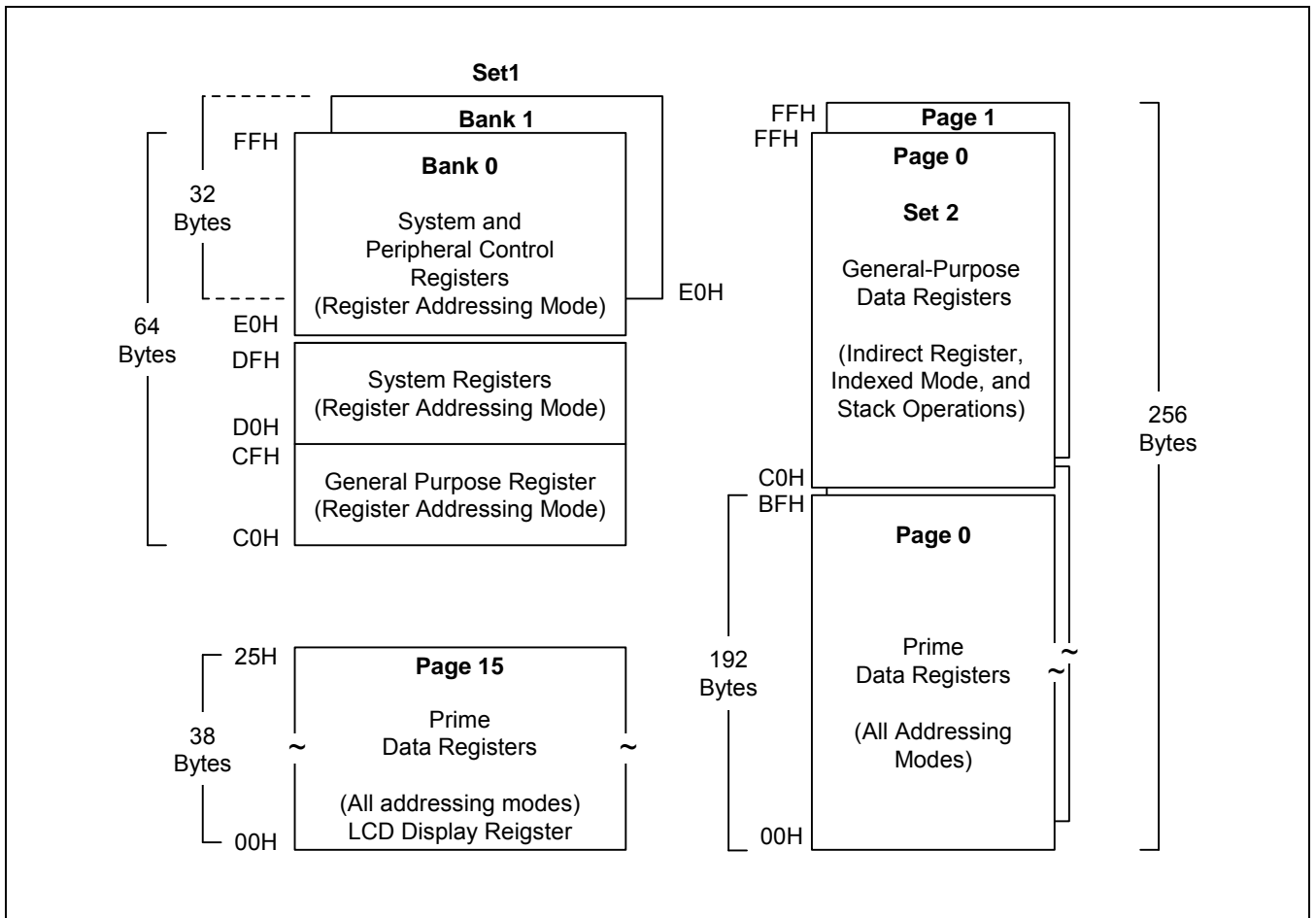


图 2-5 内部寄存器卷的地址空间 (S3C8285X/F8285X)

## 2.5 寄存器页面指针(PP)

S3F8- 系列 MCU 支持对物理上 256 字节的内部寄存器页进行逻辑扩展（通过 8 位数据总线），最多可实现16个可独立寻址的寄存器页。页面寻址由寄存器页面指针（PP，地址：DFH）来控制。S3C828BX/C8289X/C8285X中将LCD数据寄存器扩展成页文件，所以访问其他页面时要改变寄存器页面指针。

复位之后，页面指针的源页（低四位）和目标页（高四位）数值缺省为“0000B”，自动选择页面 0 作为源和目标页来进行寄存器寻址。

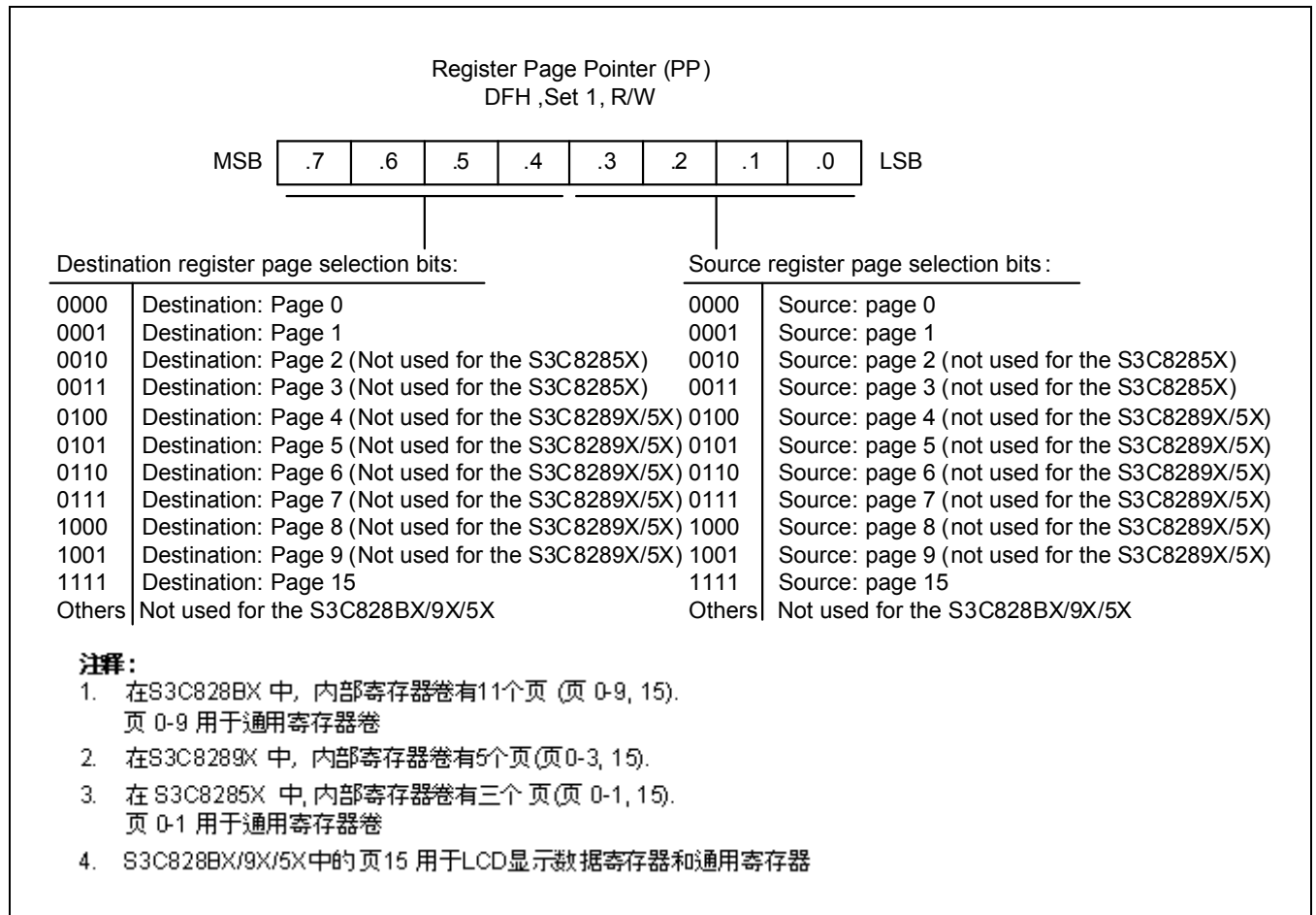


图 2-6 寄存器页面指针 (PP)

## 编程实例 2-1 在清RAM时使用页面指针 (Page 0, Page 1)

	LD	PP, #00H	;	目标 ← 0, 源 ← 0
	SRP	#0C0H		
	LD	R0, #0FFH	;	开始对页 0 进行 RAM 清零
RAMCL0	CLR	@R0		
	DJNZ	R0, RAMCL0		
	CLR	@R0	;	R0 = 00H
	LD	PP, #10H	;	目标 ← 1, 源 ← 0
	LD	R0, #0FFH	;	开始对页 1 进行 RAM 清零
RAMCL1	CLR	@R0		
	DJNZ	R0, RAMCL1		
	CLR	@R0	;	R0 = 00H

注释: DJNZ 指令的用法详见第 6-39 页。

### 2.5.1 寄存器SET 1

Set 1 指的是寄存器卷中的高 64 字节，地址为 C0H-FFH。

这 64 字节空间(E0H-FFH)的高 32 字节又被扩展为两个 32 字节的寄存器块，bank 0 和 bank 1。通过指令 SB0 或 SB1 来访问 bank 0 或 bank 1。硬件复位后默认选择 bank 0进行寻址。

Set 1(E0H-FFH)的两个高 32 字节区域 (bank0 和 bank1) 包含48个系统和外设控制寄存器，低 32 字节区域包含16 个系统寄存器 (D0H-DFH) 和 16 字节的通用工作寄存器 (C0H-CFH)。在其它寄存器空间执行的数据操作，也可以通过工作寄存器。

Set 1 中的寄存器可以随时通过寄存器寻址模式进行访问。16 字节的工作寄存器区域，则只能使用工作寄存器寻址模式 (更多关于工作寄存器寻址的信息，请查阅第 3 章,“寻址方式”)。

### 2.5.2 寄存器SET 2

对 set 1 的 64 字节地址空间 (C0H-FFH) 进行逻辑复制，以增加额外的 64 字节寄存器空间。这个扩展的空间被称为 set 2。在 S3C828BX, 中， page 0-9 可对 set 2 的地址空间 (C0H-FFH) 进行寻址。

在 S3C8289X, 中， page 0-3 可对 set 2 的地址空间 (C0H-FFH) 进行寻址。

在 S3C8285X, 中， page 0-1 可对 set 2 的地址空间 (C0H-FFH) 进行寻址。

通过寻址模式的限制，得以实现 set 1 和 set 2 的逻辑分割。Set 1 只支持寄存器寻址模式，而对 set 2 的寻址只能采用寄存器间接寻址模式或偏址寻址模式。

Set 2 寄存器区常用作堆栈操作。



### 2.5.3 主寄存器区

S3C828BX/C8289X/C8285X 有10个，4个或2个寄存器页，每页有256个字节，其中低192字节 (00H–BFH) 称为主寄存器区。主寄存器区可通过七种寻址模式中的任何一种进行访问（详见第 3 章，“寻址方式”）。

第0页的主寄存器区在复位之后就寻址。为访问第0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 或 15页的主寄存器区，必须对寄存器页面指针（PP）的源和目标进行正确设置。

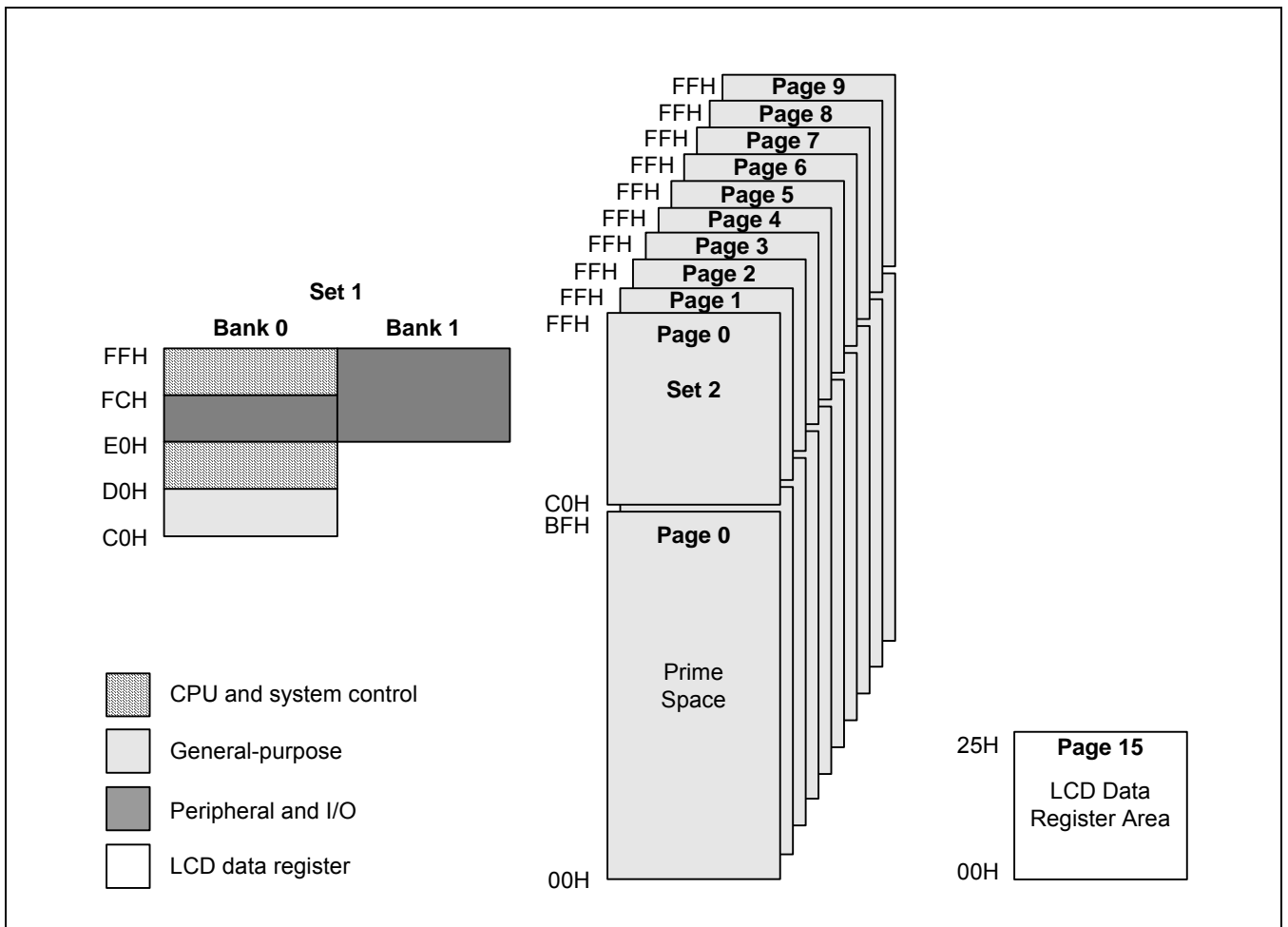


图 2-7 Set 1, Set 2, 主寄存器区和 LCD 数据寄存器

## 2.5.4 工作寄存器

指令可通过 4 位或 8 位地址来访问指定的 8 位寄存器或者 16 位寄存器对。当使用 4 位工作寄存器寻址模式时，256 字节的寄存器卷可以被看成是由 32 个 8 字节寄存器组或“片 (slice)”组成的。每个片包含 8 个 8 位寄存器。

通过两个 8 位寄存器指针 RP1 和 RP0，可以随时选择两个工作寄存器片组成一个 16 字节的工作寄存器块。使用这两个指针，可以将 16 字节的寄存器块移动到除 set 2 以外的任何可寻址空间。

在本手册中，术语“片(slice)”和“块(block)”用来帮助理解工作寄存器空间的大小和相对位置：

- 一个工作寄存器“片”是 8 个字节（8 个 8 位寄存器，R0-R7 或 R8-R15）
- 一个工作寄存器“块”是 16 个字节（16 个 8 位寄存器，R0-R15）

所有 8 字节的工作寄存器“片”中，寄存器地址的高 5 位数值完全相同。

这使得各个寄存器指针都可以指向寄存器卷中除 set 2 以外的 32 个寄存器“片”中任何一个。寄存器指针 RP0 和 RP1 中存放的是选定的两个 8 位寄存器“片”的起始地址。

系统复位后，RP0 和 RP1 总是指向 Set 1 中的 16 字节公共空间（C0H-CFH）。

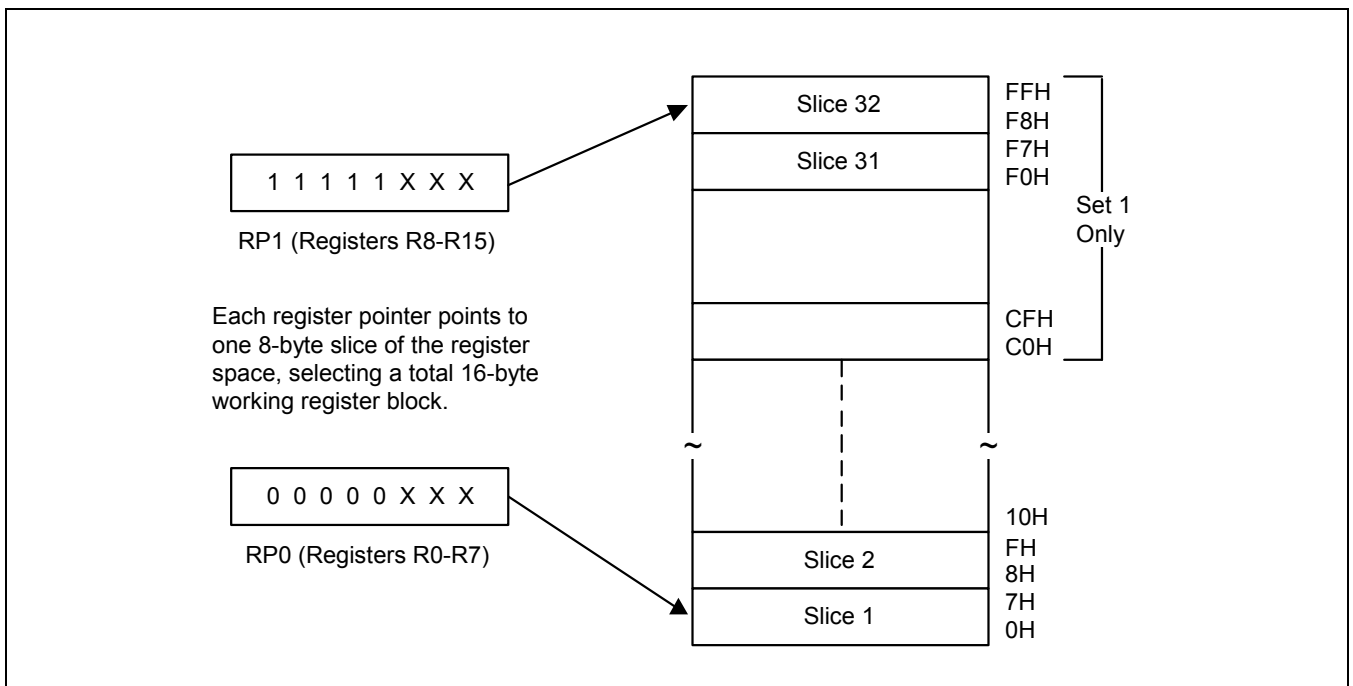


图 2-8 8字节工作寄存器区 (Slices)

### 2.5.5 使用寄存器指针

寄存器指针 RP0 和 RP1 (地址: D6H 和 D7H, Set 1), 用于选择寄存器卷中两个可移动的 8 字节工作寄存器片。复位后, RP# 指向工作寄存器的公共空间: RP0 指向地址 C0H-C7H, RP1 指向 C8H-CFH。

通过 SRP 或 LD 指令可以改变寄存器指针 RP0/ RP1 的值 (如 [图 2-9](#) 和 [图 2-10](#))。

用工作寄存器寻址方式, 只能访问当前 RP0 和 RP1 指定的两个 8 字节的工作寄存器片。但不能用寄存器指针来选择 set 2 (C0-FFH) 中的工作寄存器, 因为这些地址空间只支持间接寄存器寻址模式和偏址寻址模式。

16 字节的工作寄存器块通常由两个相邻的 8 字节工作寄存器片组成。编程时, 一般建议将 RP0 指向“低”地址的片, 而 RP1 指向“高”地址的片 (如 [图 2-9](#))。某些情况下, 可能需要将工作寄存器定义在不同的 (不相邻的) 寄存器空间中 (如 [图 2-10](#)), RP0 指向“高”地址的片, 而 RP1 指向“低”地址的片。

由于寄存器指针可以指向两个工作寄存器片中的任意一个, 用户可根据程序需求灵活定义工作寄存器空间。

#### 编程实例 2-2 设置寄存器指针

SRP	#70H	;	RP0	←	70H, RP1	←	78H
SRP1	#48H	;	RP0	←	no change, RP1	←	48H,
SRP0	#0A0H	;	RP0	←	A0H, RP1	←	no change
CLR	RP0	;	RP0	←	00H, RP1	←	no change
LD	RP1, #0F8H	;	RP0	←	no change, RP1	←	0F8H

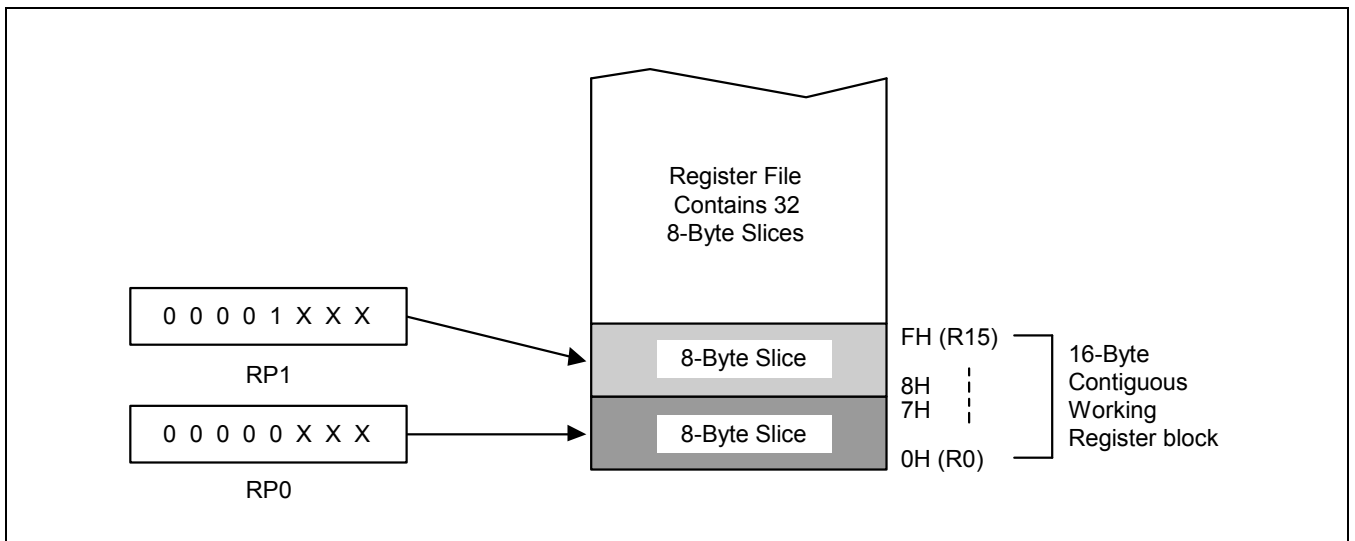


图 2-9 相邻的16字节工作寄存器块

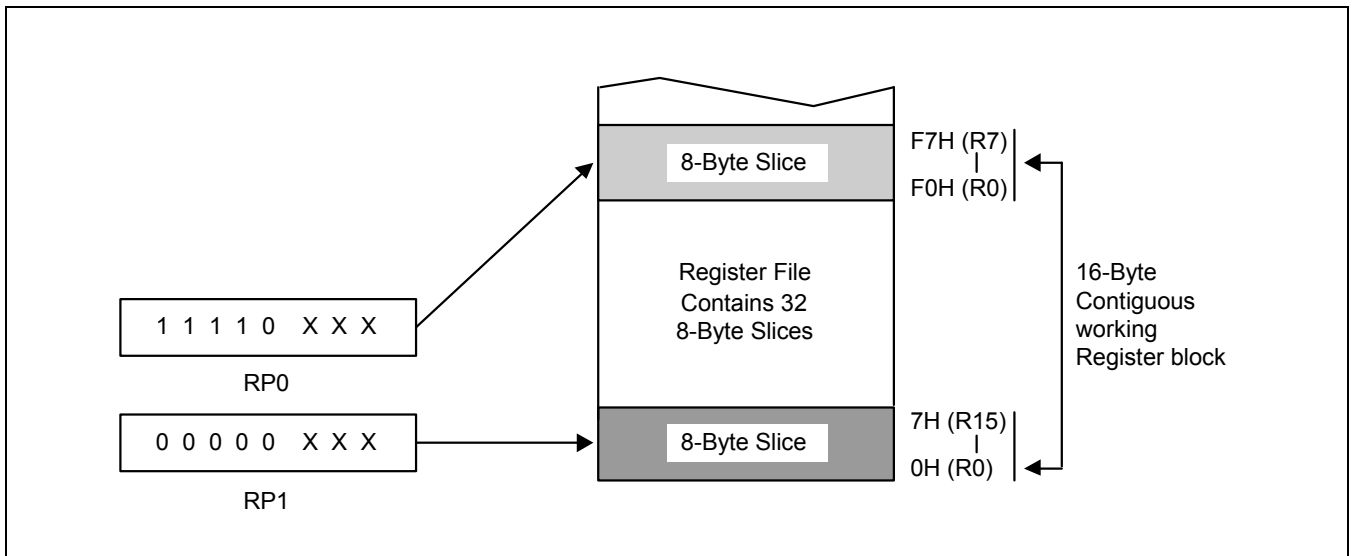


图 2-10 非相邻的 16 字节工作寄存器块

## 编程实例 2-3 使用 RP 对寄存器 80H-85H 的内容求和

用寄存器指针来对寄存器 80H-85H 中的内容求和。寄存器 80H-85H 中的存放数据分别为 10H, 11H, 12H, 13H, 14H和15H:

```
SRP0 #80H      ; RP0 ← 80H
ADD  R0,R1     ; R0 ← R0 + R1
ADC  R0,R2     ; R0 ← R0 + R2 + C
ADC  R0,R3     ; R0 ← R0 + R3 + C
ADC  R0,R4     ; R0 ← R0 + R4 + C
ADC  R0,R5     ; R0 ← R0 + R5 + C
```

6 个寄存器的和 (6FH) 存放在 R0 (80H) 中。例子中的指令共占用 12 字节的代码长度, 执行时间为 36 个时钟周期。如果不用寄存器指针, 那就得按照下面的指令顺序来做:

```
ADD  80H,81H   ; 80H ← (80H) + (81H)
ADC  80H,82H   ; 80H ← (80H) + (82H) + C
ADC  80H,83H   ; 80H ← (80H) + (83H) + C
ADC  80H,84H   ; 80H ← (80H) + (84H) + C
ADC  80H,85H   ; 80H ← (80H) + (85H) + C
```

现在, 6 个寄存器的和依然放在 80H 当中, 但是所有指令总共占用了 15 字节的代码长度而非 12 字节, 且执行时间为 50 个时钟周期而非 36 个。

## 2.6 寄存器寻址

S3C8- 系列单片机的寄存器结构提供了一种效率高的工作寄存器寻址方式，该方式充分利用了短指令格式以缩短程序执行时间。

在寄存器寻址模式中，操作数存放在某个特定的寄存器或寄存器对中。用寄存器寻址模式可以访问寄存器空间中除 **set 2** 以外的的任何地址。使用工作寄存器寻址时，通过寄存器指针指定一个 8 字节的工作寄存器空间和该空间内的一个 8 位寄存器。

寄存器寻址时，既可以视其为单独的 8 位寄存器，也可以视为成对的 16 位寄存器空间。在 16 位寄存器对中，第一个 8 位寄存器的地址总是偶数，而另一个寄存器地址则是奇数。16 位数据的高位字节存放在偶地址寄存器中，低位字节存放在邻近的 (+1) 奇地址寄存器中。

工作寄存器寻址模式与寄存器寻址模式的不同之处在于，它通过寄存器指针来指定一个 8 字节工作寄存器空间，以及其中的某个 8 位工作寄存器。

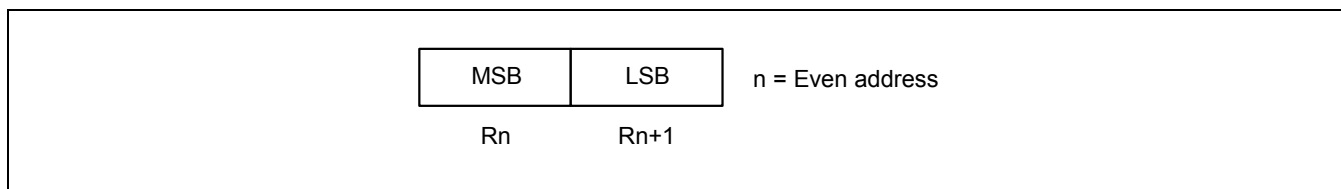


图 2-11 16位工作寄存器结构

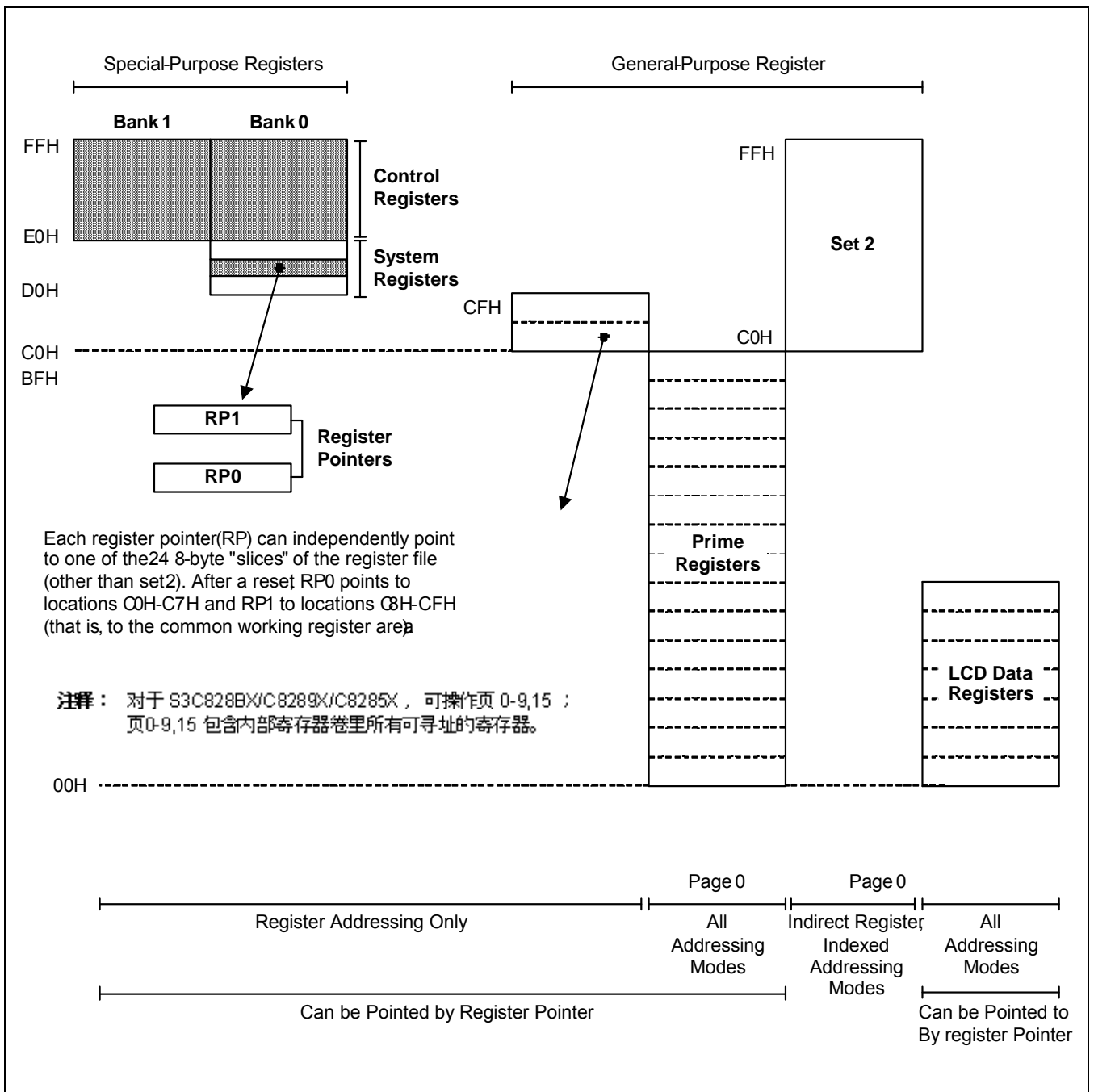


图 2-12 寄存器卷寻址模式

2.6.1 通用工作寄存器区 (C0H-CFH)

系统复位后，寄存器指针 RP0 和 RP1 自动指向 set 1中两个 8 字节的寄存器片(C0H-CFH)，组成 16 字节的寄存器块：

RP0	C0H-C7H
RP1	C8H-CFH

这 16 字节的地址空间称为通用工作寄存器区。就是说无论当前操作所要访问的是哪个页面，都可以把该空间的寄存器作为工作寄存器使用。典型地，在不同页面间进行数据交换操作时，可使用工作寄存器作为临时存储。

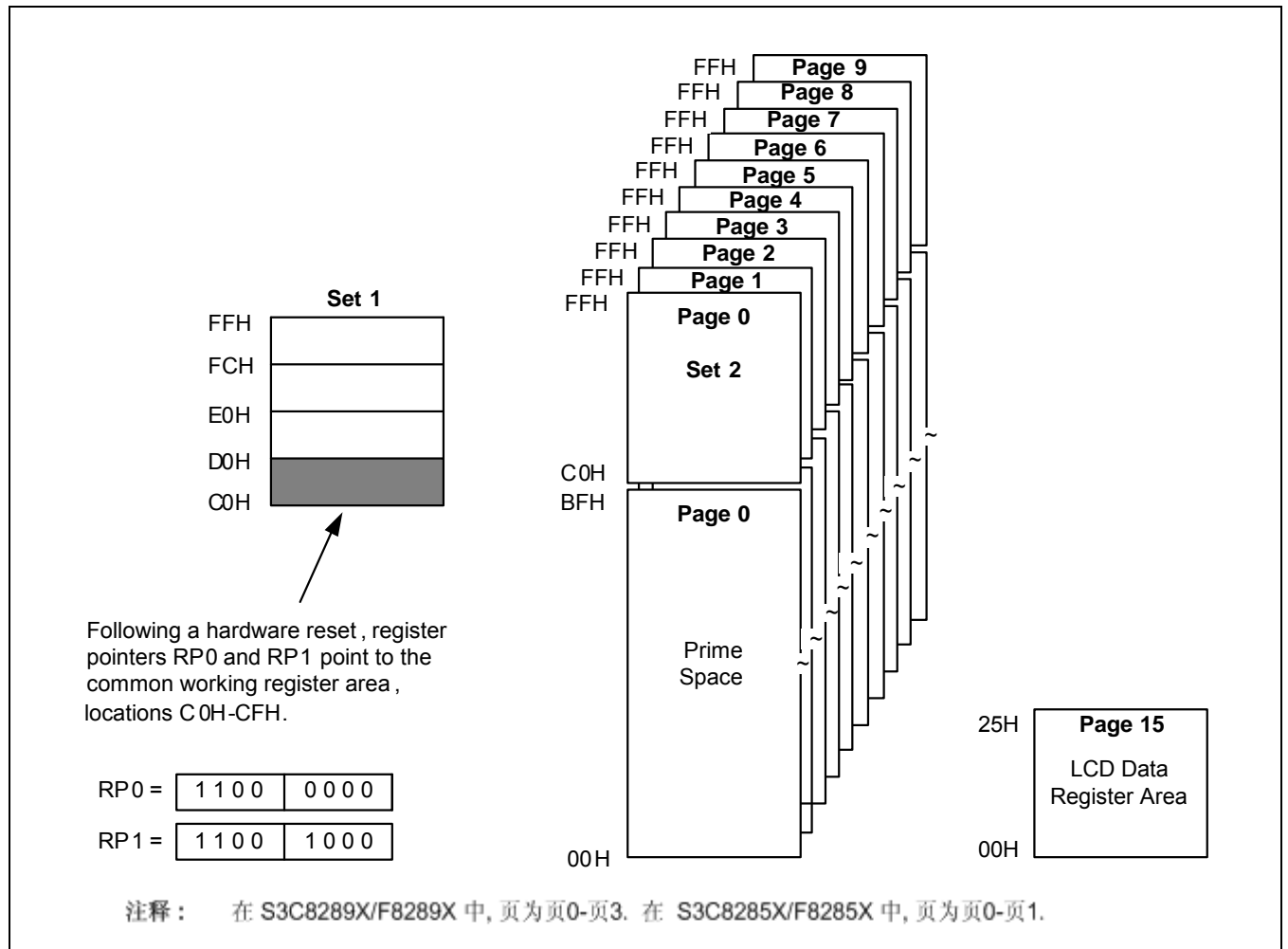


图 2-13 通用工作寄存器区

## 编程实例 2-4 访问通用工作寄存器区

如下例所示，要访问通用工作寄存器 (C0H-CFH)，只能采用工作寄存器寻址模式：

例1:           LD     0C2H, 40H                 ;     非法寻址模式！

改为工作寄存器寻址模式：

```

SRP   #0C0H
LD     R2, 40H                 ;     R2 (C2H) ← 地址 40H 中存放的值

```

例2:

          ADD    0C3H, #45H                 ;     非法寻址模式！

改为工作寄存器寻址模式：

```

SRP   #0C0H
ADD    R3, #45H                 ;     R3 (C3H) ← R3 + 45H

```

## 2.6.2 4 位工作寄存器寻址方式

每个寄存器指针定义了一个可移动的 8 字节寄存器片，其中的地址信息作为一扇寻址的“窗”，使得指令只须 4 位地址就可以实现对工作寄存器的有效访问。在工作寄存器寻址时，8 位地址是采用下述方法构成的：

- 4 位地址的最高位选择一个寄存器指针（“0”选择 RP0，“1”选择 RP1）
- 寄存器指针的高 5 位选择寄存器卷中的某个 8 字节寄存器片
- 指令中 4 位地址的低 3 位选定 8 字节寄存器片中的一个

如 [图 2-14](#)，操作的结果是，寄存器指针的高 5 位和指令地址的低 3 位一起组成完整的 8 位寄存器地址。只要寄存器指针中保存的地址不变，指令中 4 位地址的低 3 位总是指向同一个 8 字节寄存器片。

[图 2-15](#) 给出了一个典型的 4 位工作寄存器寻址实例。指令“INC R6”的最高位是“0”，这将选择 RP0。RP0 的高 5 位（01110B）与指令中 4 位地址的低 3 位（110B）一起组成了寄存器地址 76H（01110110B）。



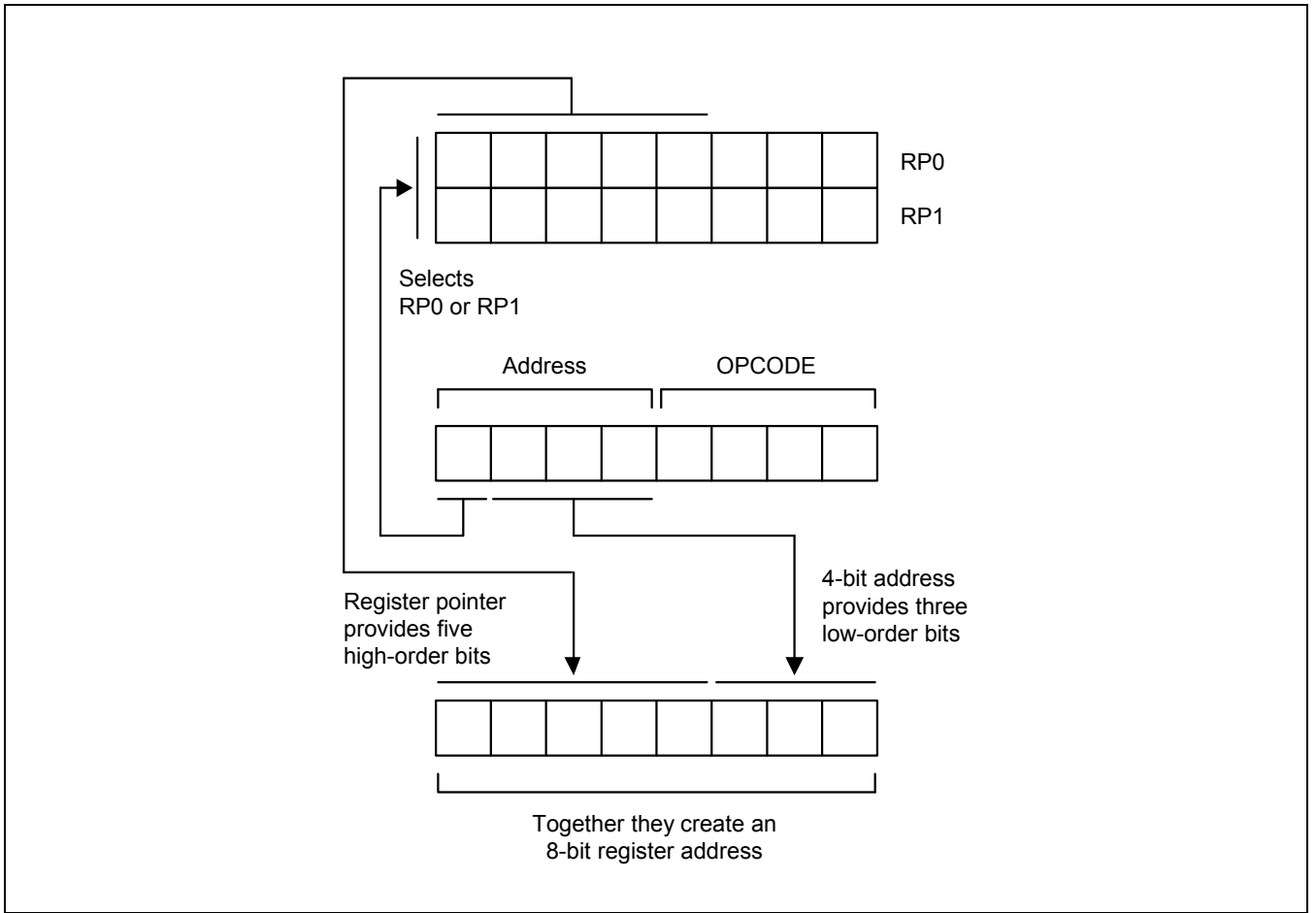


图 2-14 4位工作寄存器寻址方式

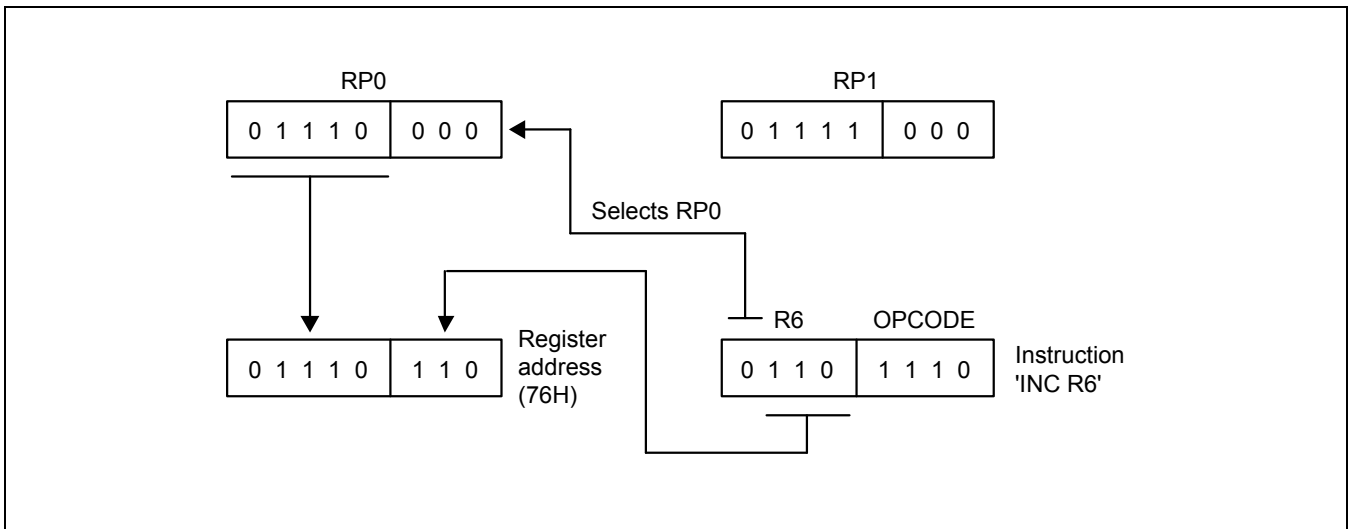


图 2-15 4位工作寄存器寻址方式实例

### 2.6.3 8 位工作寄存器寻址方式

还可以通过 8 位工作寄存器寻址方式来访问工作寄存器区。首先，指令地址的高 4 位必须是“1100B”，这 4 位数据 (1100B) 表示余下的 4 位与 4 位工作寄存器寻址方式相同。

如图 2-16，8 位地址的低阶位形成机制与 4 位工作寄存器寻址时类似：第 3 位选择 RP0 或 RP1，用来产生最终地址的高 5 位；而最终地址的低 3 位则由原始指令提供。

图 2-17 给出了一个典型的 8 位工作寄存器寻址实例。指令地址的高 4 位 (1100B) 表明寻址方式为 8 位寄存器寻址。第 3 位 (“1”) 选择 RP1，所以 RP1 中的高 5 位 (10101B) 成为寄存器地址的高 5 位，寄存器地址的低 3 位 (011B) 则由 8 位指令地址的低 3 位提供。RP1 中的 5 个地址位和指令中的 3 个地址位组合，形成了完整的寄存器地址，0ABH (10101011B)。

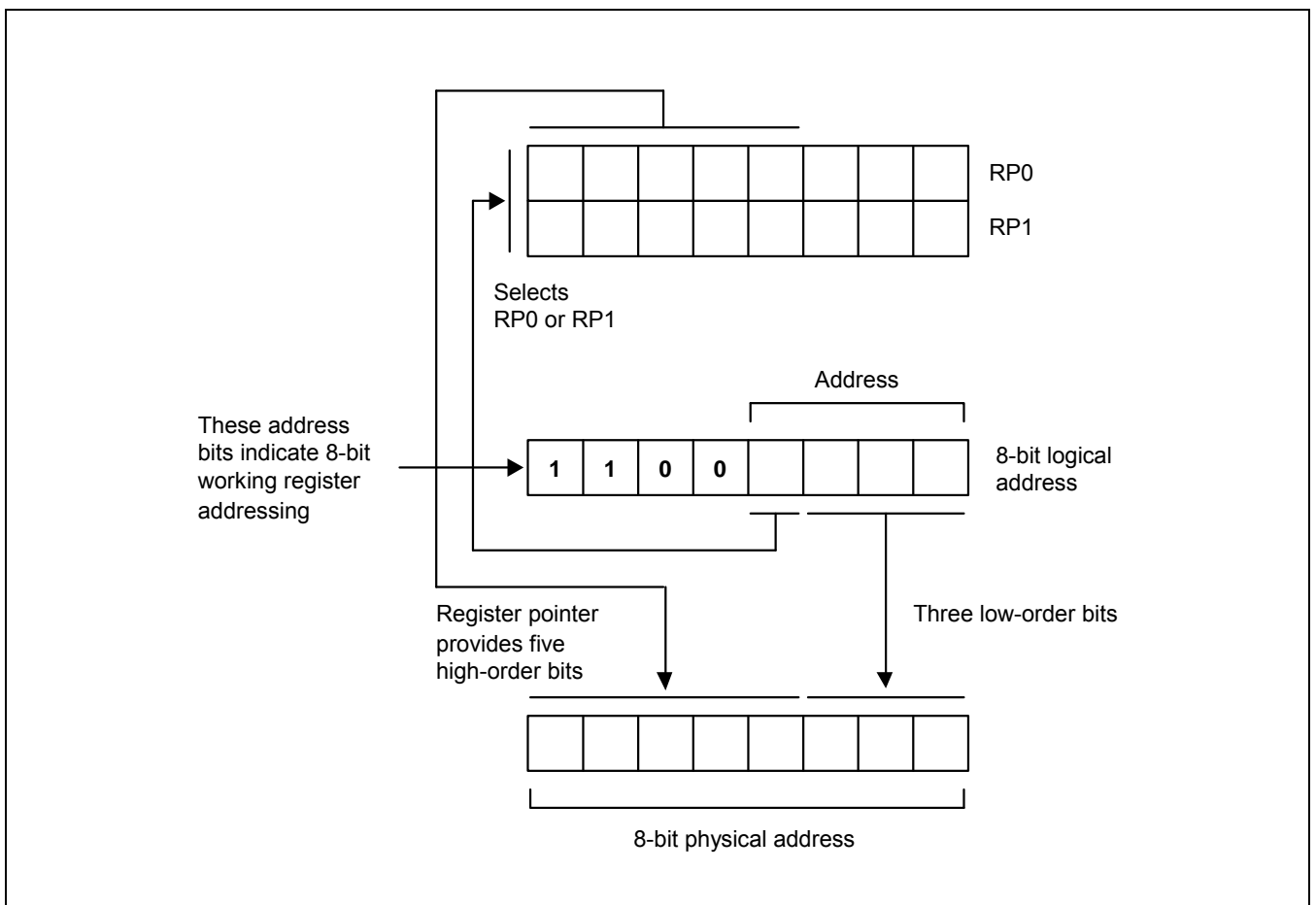


图 2-16 8位工作寄存器寻址方式

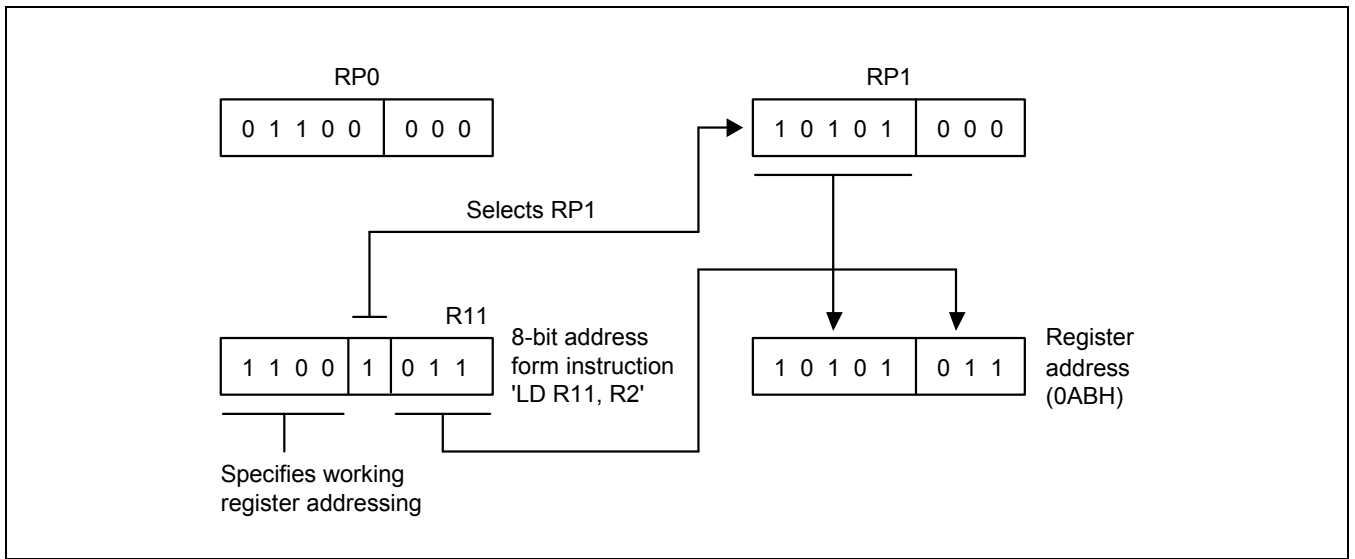


图 2-17 8位工作寄存器寻址方式实例

## 2.7 系统和用户栈

S3C8- 系列单片机通过栈来处理数据存储，子程序调用和返回。PUSH 和 POP 指令用于控制栈操作。S3C828BX/C8289X/C8285X 支持在内部寄存器卷中进行栈操作。

### 2.7.1 栈操作

栈用来储存程序调用以及中断的返回地址，也用来储存数据。CALL 指令将 PC 的值压入栈，而 RET 指令将其从栈中弹出。当中断发生时，PC 和 FLAGS 寄存器的值被压入栈，指令 IRET 则将这些值弹出到它们原来的地址。栈指针总是在 PUSH 操作之前先减“1”，在 POP 操作之后加“1”。栈指针(SP)总是指向栈的顶端，如 [图 2-18](#) 所示。

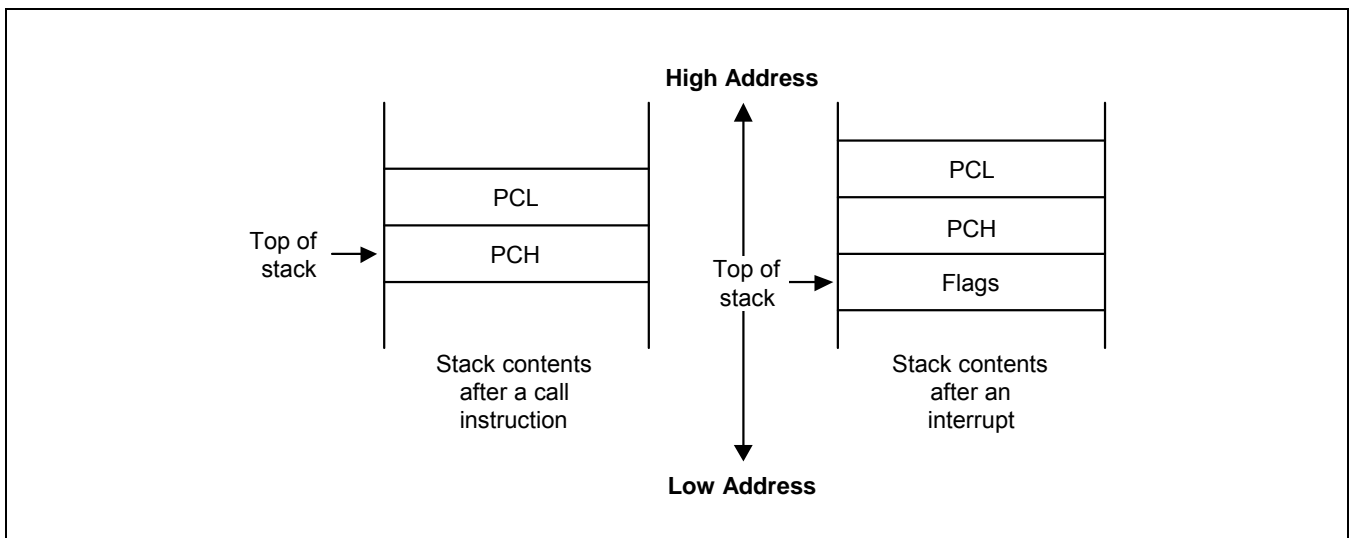


图 2-18 栈操作

### 2.7.2 用户自定义栈

可以在内部寄存器卷中堆栈自定义栈数据储存区域。指令 PUSHUI, PUSHUD, POPUI, POPUD 支持用户自定义栈操作。

### 2.7.3 栈指针 (SPL, SPH)

寄存器地址 D8H 和 D9H 存放用于栈操作的 16 位栈指针(SP)。SP 的高字节地址，SP15–SP8，存放在 SPH 寄存器 (D8H) 中；低字节地址，SP7–SP0，存放在 SPL 寄存器 (D9H) 中。系统复位后，SP 的值不确定。

由于 S3C828BX/C8289X/C8285X 只有内部存储空间，SPL 必须被初始化成介于 00H-FFH 之间的 8 位数值，而 SPH 则用不到，需要时可以把它作为通用寄存器使用。

当 SPH 寄存器用作通用数据寄存器时，如果因为 SPL 寄存器中正常的栈操作，如栈地址增加或减少而导致溢出发生，将会影响到 SPH 寄存器，并覆盖当前存储的数据。为了避免 SPH 寄存器被覆盖的情况发生，最好把 SPL 的初始值设为“FFH”而非“00H”。

## 编程实例 2-5 用 PUSH 和 POP 实现标准栈操作

下面的例子演示了在内部寄存器卷中如何通过 PUSH 和 POP 指令进行栈操作：

```

LD      SPL, #0FFH   ;      SPL      ←  FFH
                          ;      (通常 SPL 被初始化为 0FFH )
.
.
.
PUSH   PP            ;      栈地址 0FEH ←  PP
PUSH   RP0           ;      栈地址 0FDH ←  RP0
PUSH   RP1           ;      栈地址 0FCH ←  RP1
PUSH   R3            ;      栈地址 0FBH ←  R3
.
.
.
POP    R3            ;      R3        ←  栈地址 0FBH
POP    RP1           ;      RP1       ←  栈地址 0FCH
POP    RP0           ;      RP0       ←  栈地址 0FDH
POP    PP            ;      PP        ←  栈地址 0FEH

```

# 3 寻址模式

## 3.1 概述

通过程序指针(PC)读取程序存储空间的指令然后执行。指令隐含着要进行的操作和操作数。寻址模式是用于确定操作数地址的一种方法。SAM8RC 指令中的操作数可以是条件代码、立即数，或者寄存器卷、程序存储区、数据存储区的地址。

S3F8- 系列的指令集支持 7 种寻址模式，但这些寻址模式并非适用于所有指令。  
7 种寻址模式和它们的符号表示：

- 寄存器寻址模式(R)
- 间接寄存器寻址模式(IR)
- 偏址寻址模式(X)
- 直接寻址模式(DA)
- 间接寻址模式(IA)
- 相对地址寻址模式(RA)
- 立即数寻址模式(IM)

### 3.2 寄存器寻址模式(R)

寄存器寻址模式 (R) 中，操作数是具体寄存器或寄存器对中的内容，如 [图 3-1](#)。

工作寄存器寻址模式与寄存器寻址模式不同。因为工作寄存器是通过一个 16 位的寄存器指针，来指定 8 字节的工作寄存器空间，和空间内的某个 8 位寄存器，如 [图 3-2](#)。

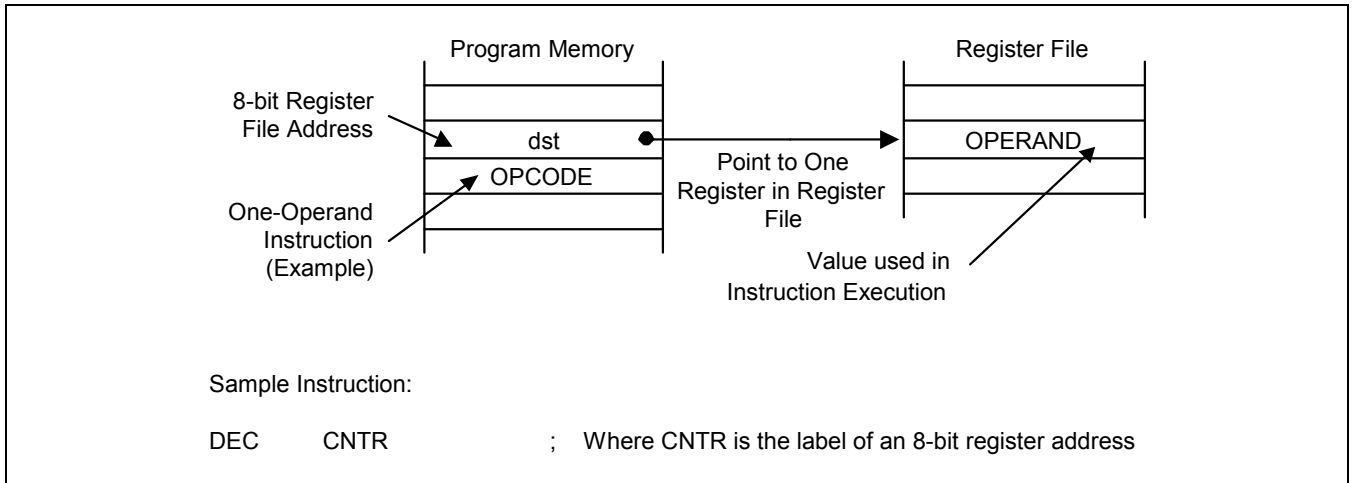


图 3-1 寄存器寻址模式

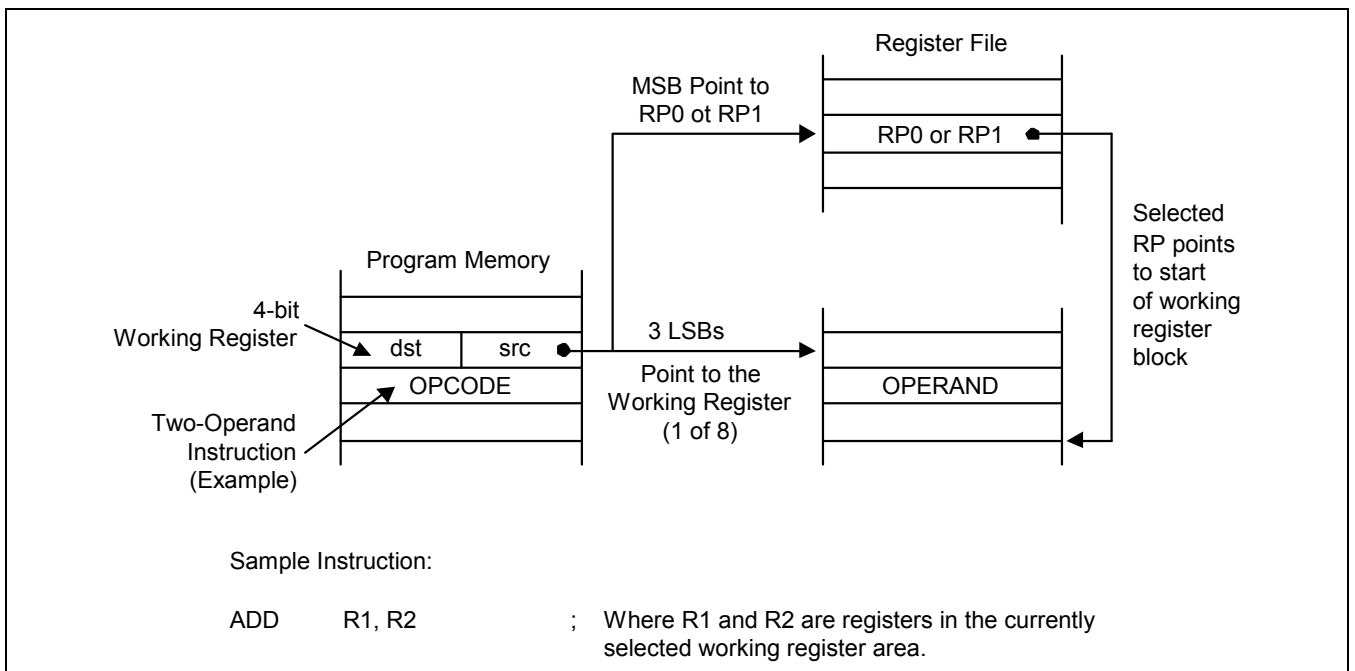


图 3-2 工作寄存器寻址模式

### 3.3 间接寄存器寻址模式(IR)

在间接寄存器寻址模式中，指定寄存器或寄存器对中存放的是操作数的地址。根据所用的指令，物理地址可能是寄存器卷中的寄存器、程序存储器（ROM），或者外部数据存储器（如 [图 3-3](#) 至 [图 3-6](#)）。

可以用任意的 8 位寄存器来访问其它的寄存器，也可以用任意的 16 位寄存器组来访问其它的存储空间。但要注意，不能通过间接寄存器寻址模式对 Set 1 中地址段 C0H-FFH 进行访问。

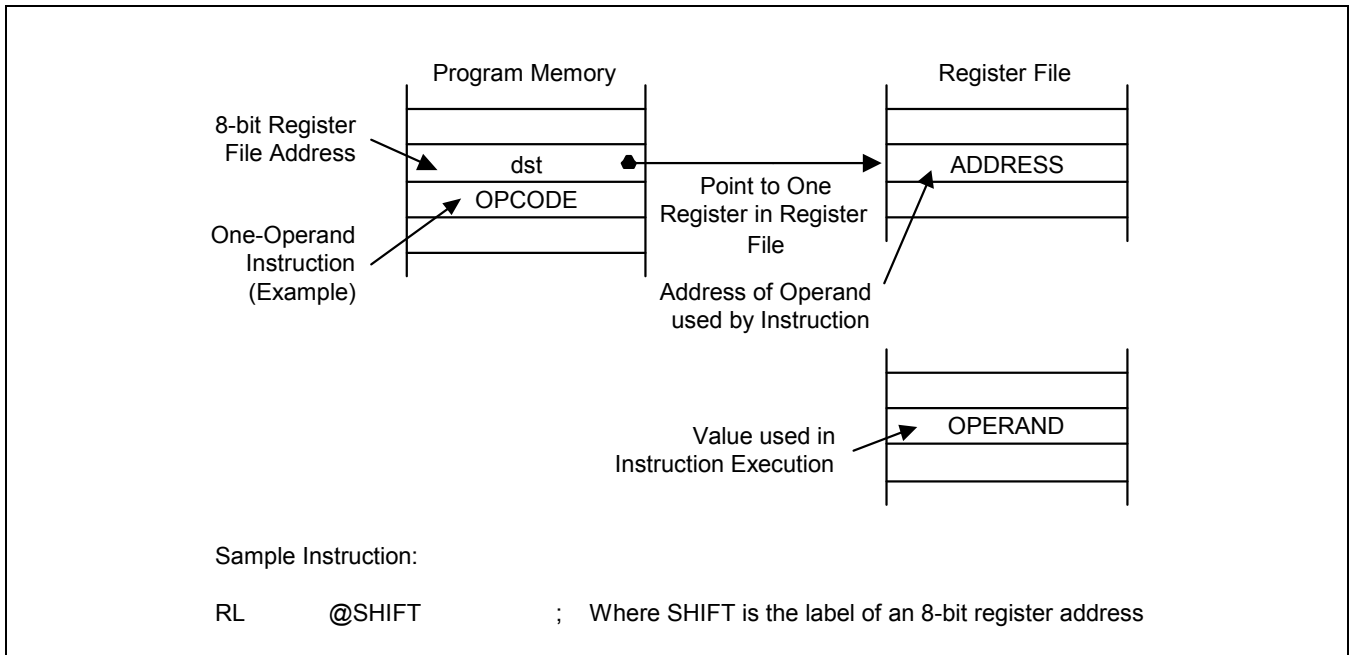


图 3-3 寄存器卷中的间接寄存器寻址模式



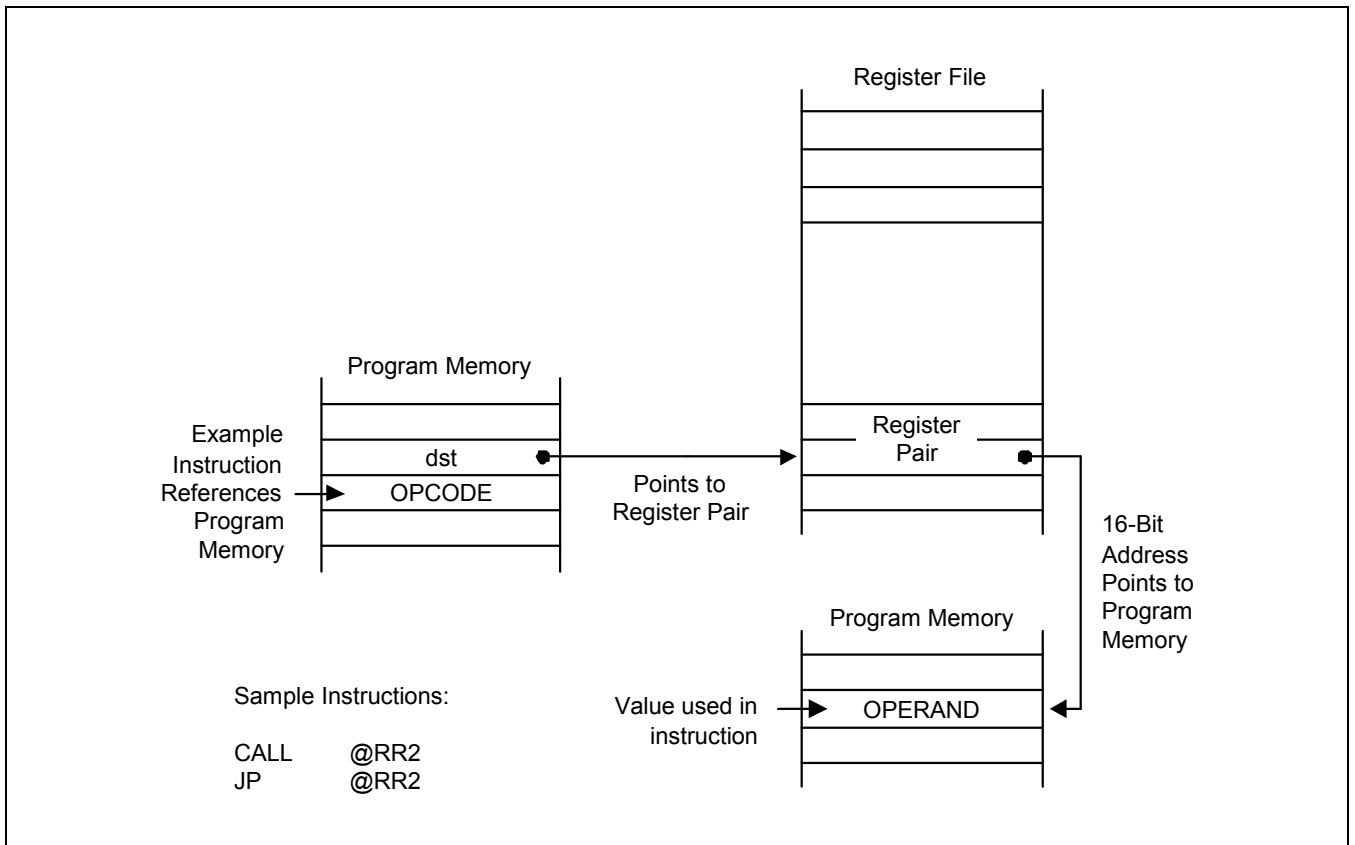


图 3-4 程序存储空间的间接寄存器寻址

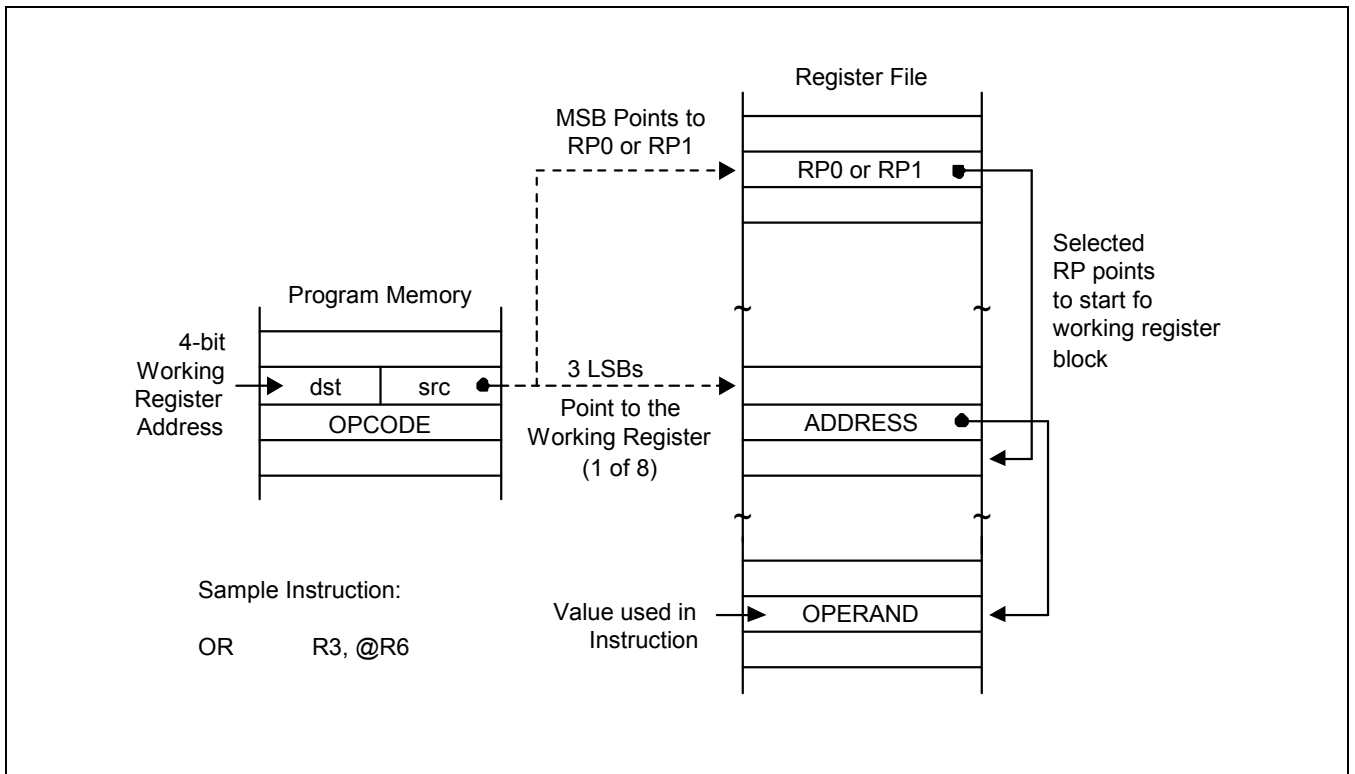


图 3-5 寄存器卷中的间接寄存器寻址

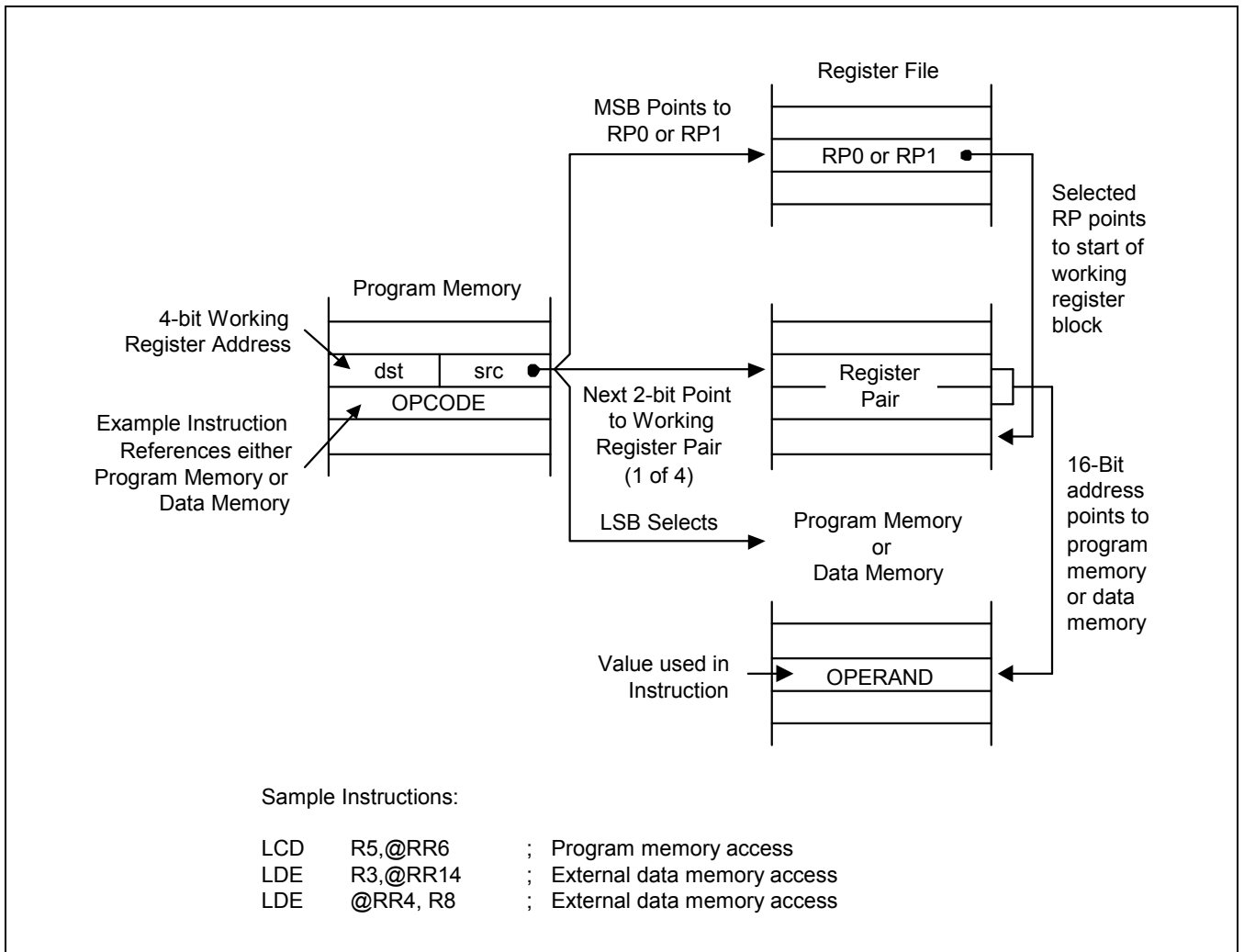


图 3-6 工作寄存器间接访问程序存储器或数据存储器

### 3.4 偏址寻址模式(X)

在指令执行时，偏址寻址模式（X）是在地址的基础上加上一个偏移地址量，计算出有效的操作数地址（如图 3-7）。偏址寻址可以用来访问内部寄存器卷或外部数据存储器空间。但要注意，不能通过它对 Set 1 中地址段 C0H-FFH 进行寻址。

在短指令寻址模式下，8 位的偏移量被认为是介于 -128 到 +127 之间的一个有符号整数，这只用于外部存储器访问（如图 3-8）。

对寄存器卷寻址时，指令提供的 8 位地址与工作寄存器中的 8 位偏移地址相加得到操作数地址。对外部存储器访问时，地址存放在指令指示的 16 位工作寄存器中，指令中给出的 8 位或 16 位偏移地址加到地址上，得到操作数地址（如图 3-9）。

支持对寄存器卷进行偏址寻址的指令只有 Load (LD)。LDC 和 LDE 支持内部程序存储器和外部数据存储器（如果存在）的偏址寻址模式。

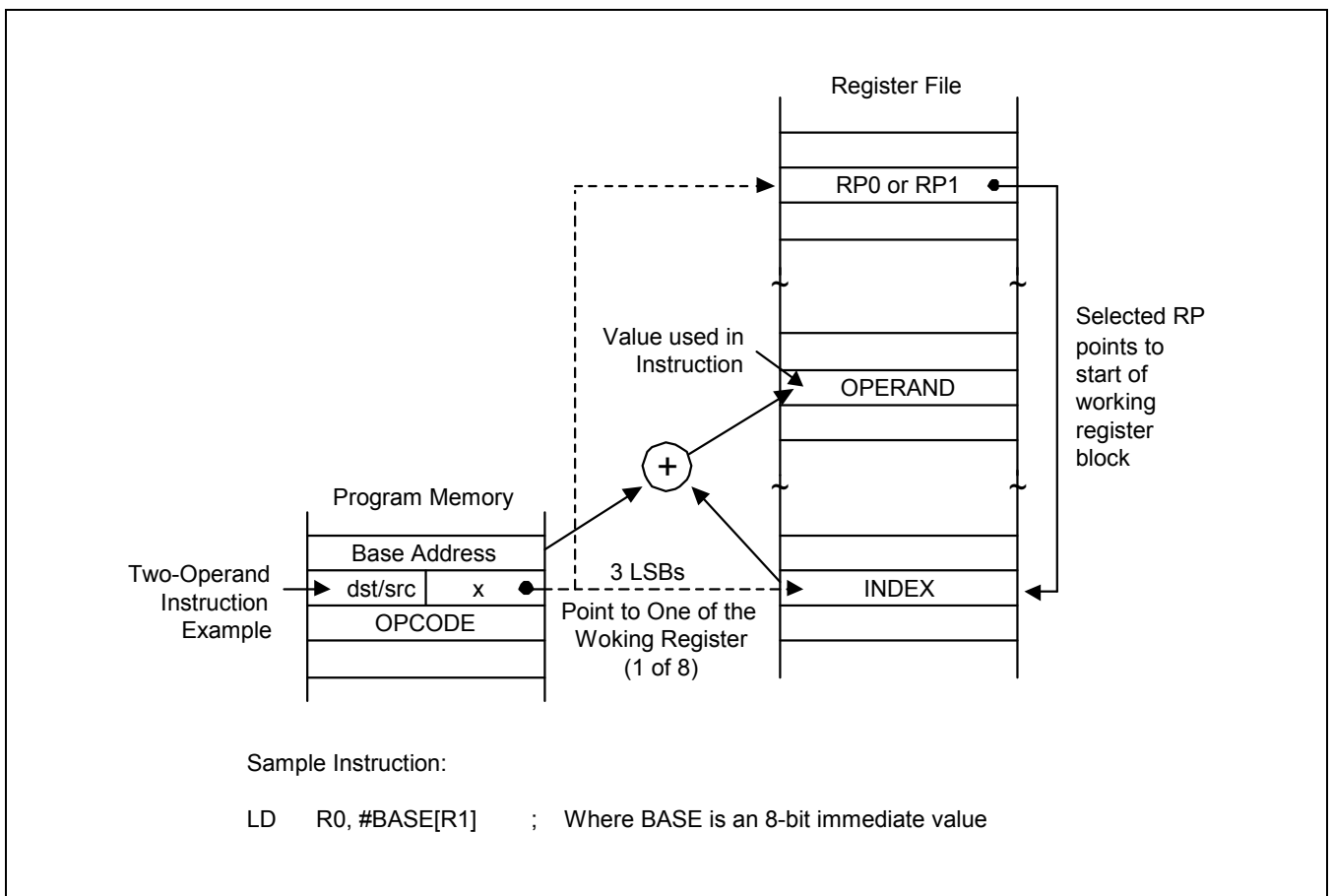


图 3-7 寄存器空间的偏址寻址模式

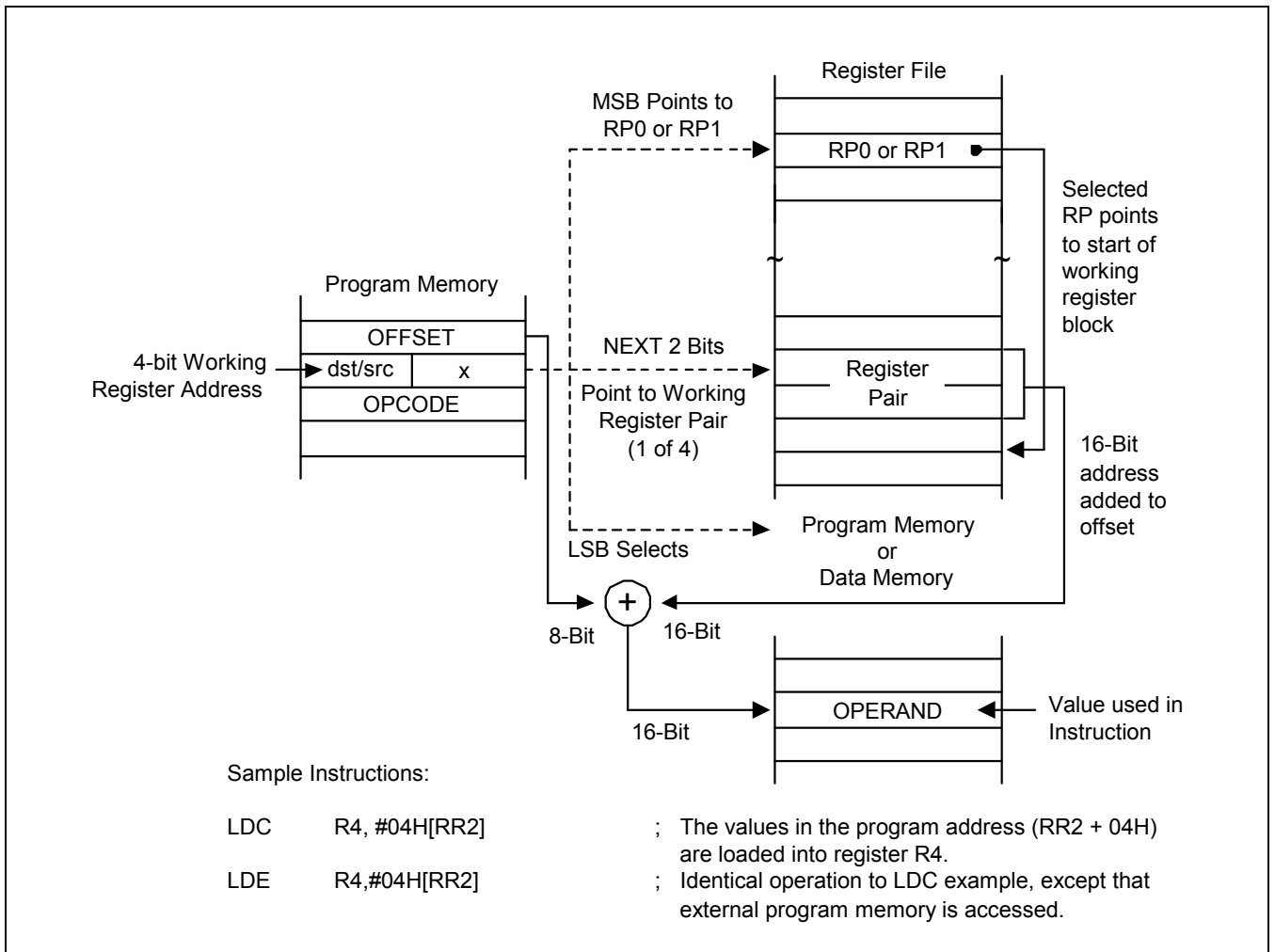


图 3-8 偏址寻址模式中短格式访问程序或数据存储空间

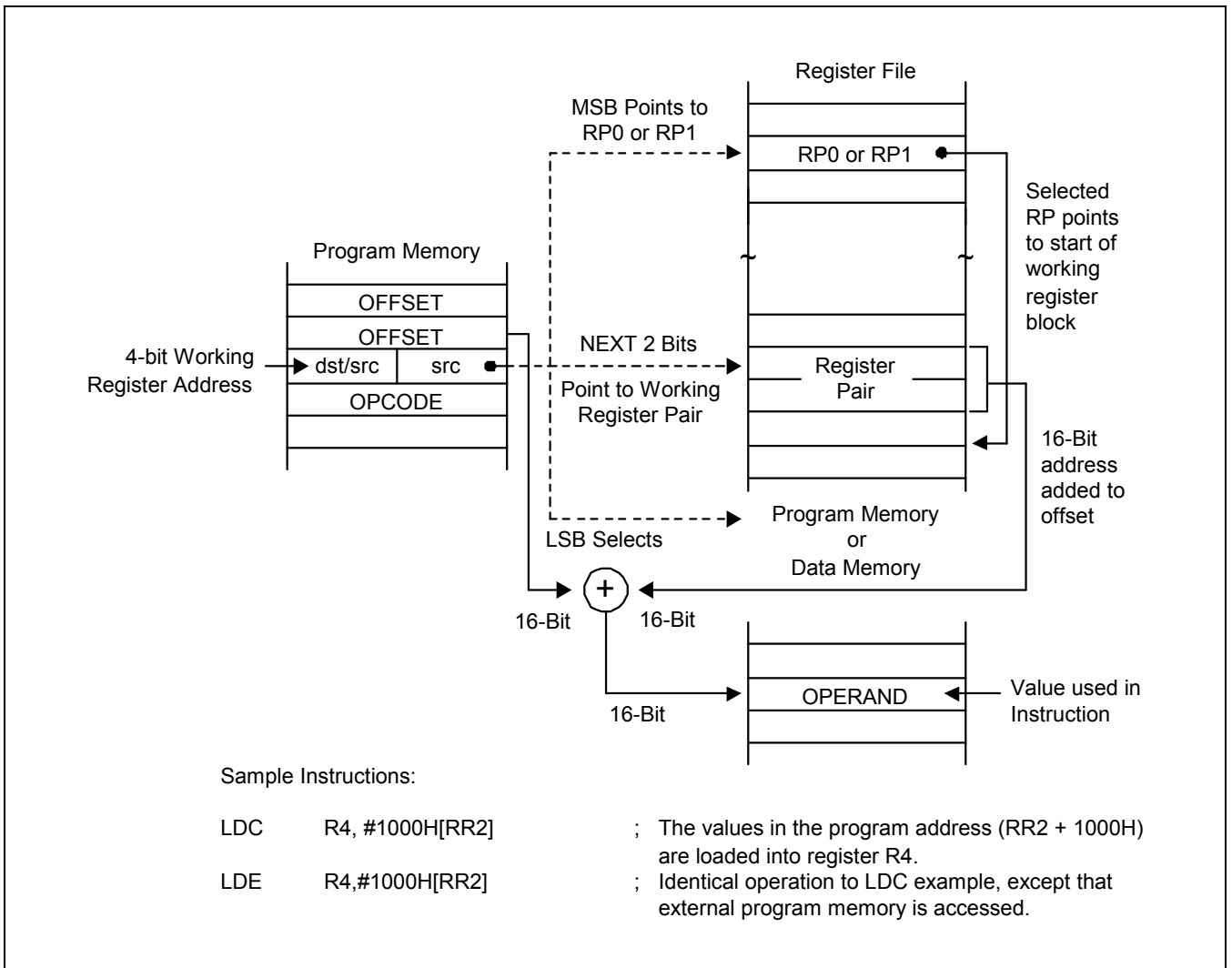


图 3-9 偏址寻址模式中长格式访问程序或数据存储空间

### 3.5 直接寻址模式(DA)

在直接寻址模式中，指令提供操作数的 16 位存储器地址。执行 Jump(JP) 和 Call(CALL) 指令时，就是采用这种寻址模式指定 16 位目标地址并将其装入PC。

LDC 和 LDE 指令即运用直接寻址模式为数据传送操作提供源地址或目标地址，LDC 访问程序存储空间，LDE 访问外部数据存储空间。

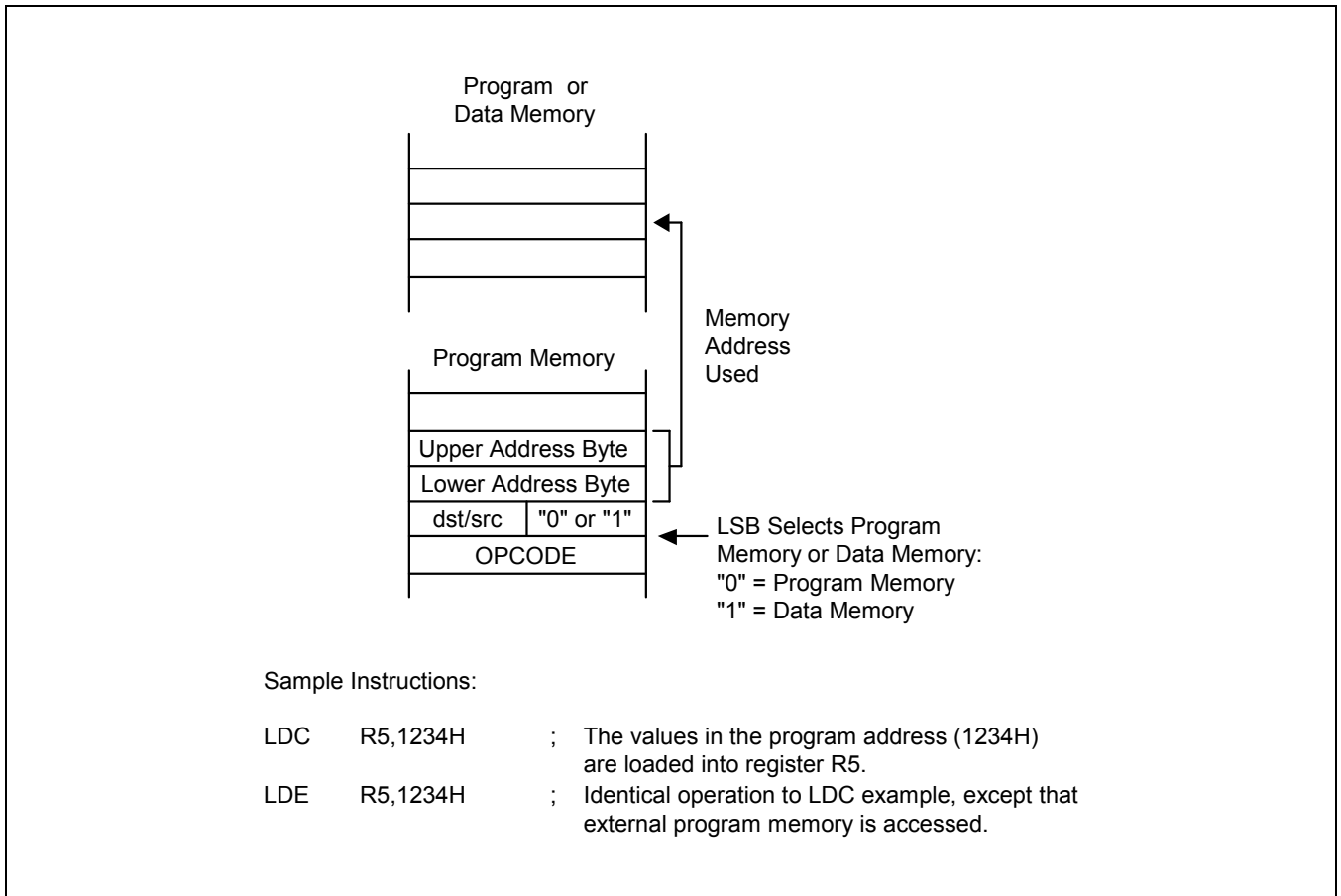


图 3-10 Load 指令的直接寻址

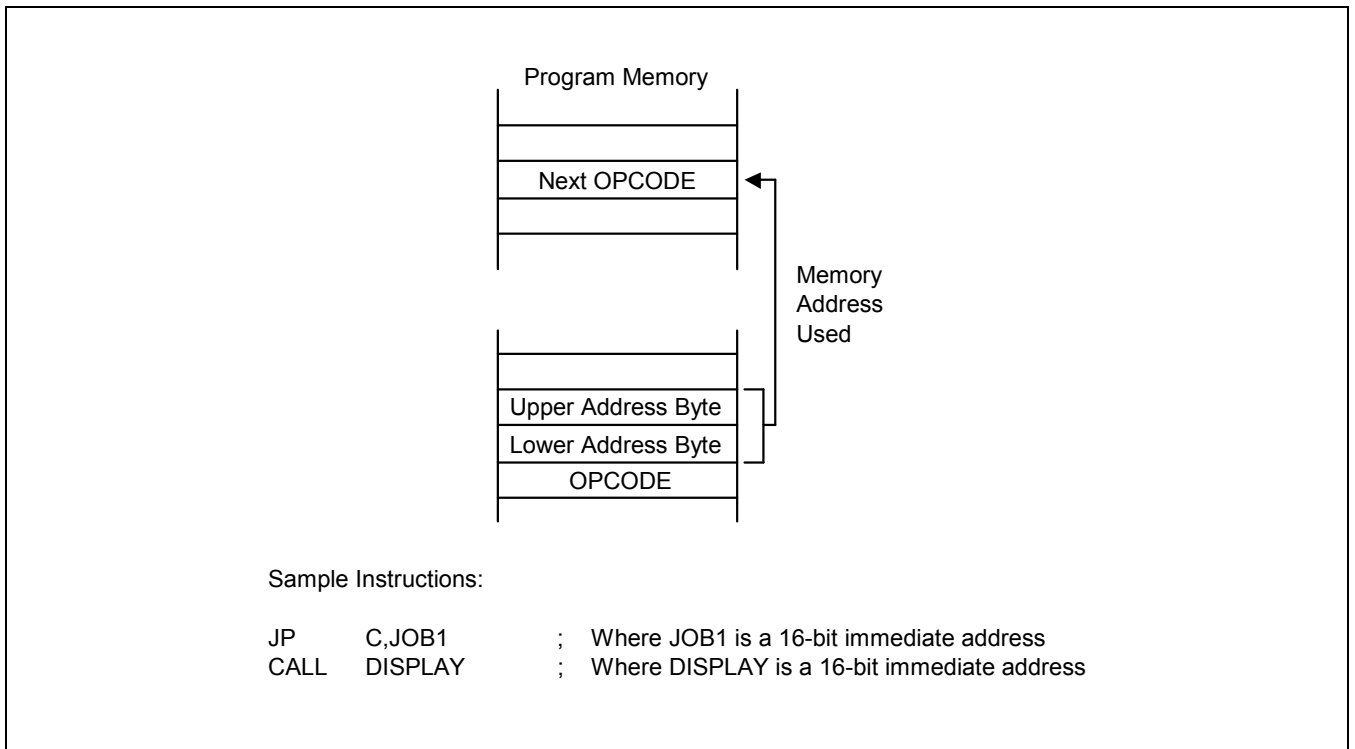


图 3-11 Call, Jump 的直接寻址



### 3.6 间接寻址模式(IA)

在间接寻址模式中，须指定程序存储空间中低 256 字节中的某个地址，其中放有要执行的下一条指令地址。只有CALL 指令支持间接寻址模式。

由于间接寻址模式规定操作数只能存放在程序存储空间的低 256 字节中，所以指令中只有一个 8 位地址；目标地址的高 8 位全部为 0。

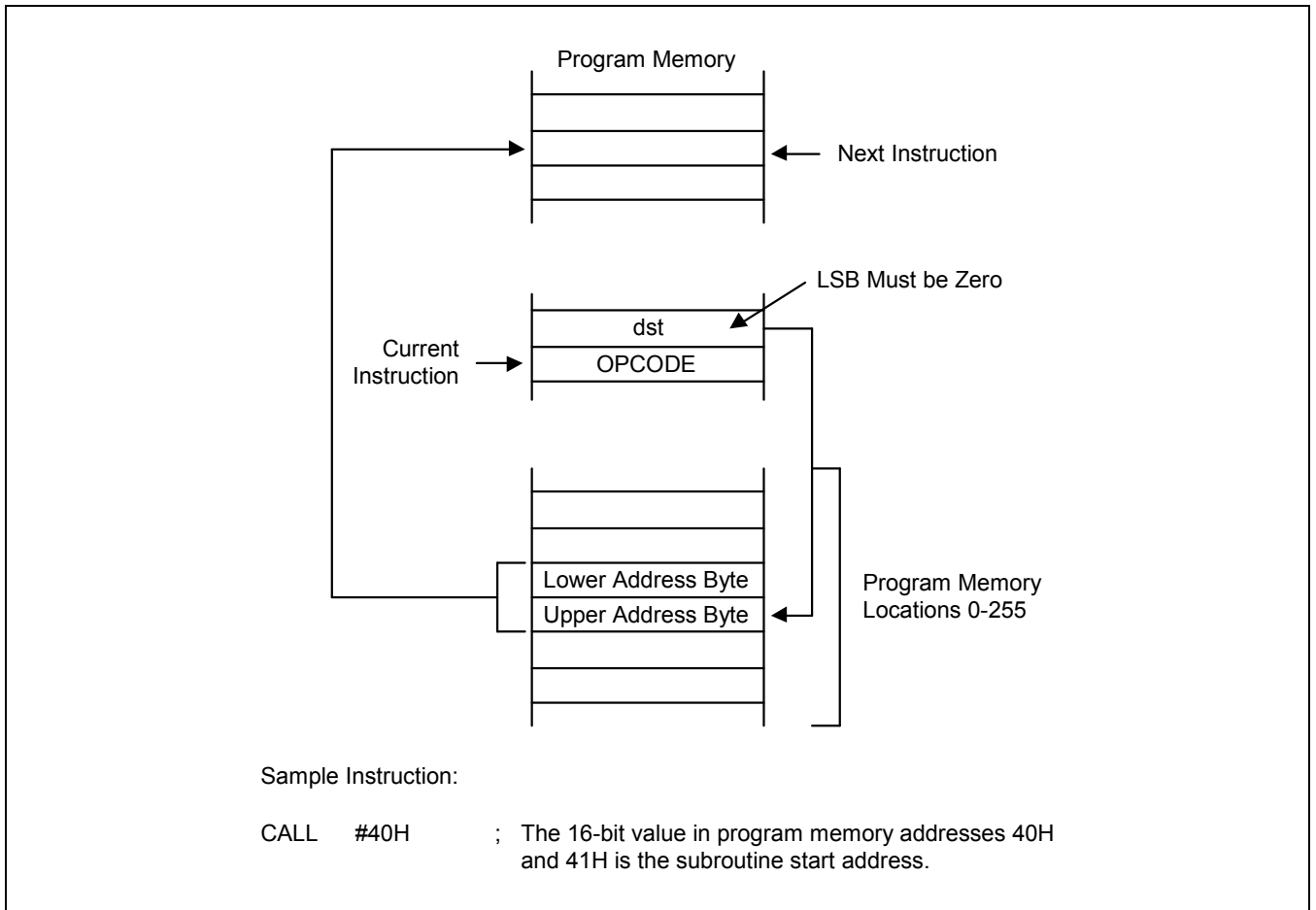


图 3-12 间接寻址模式

### 3.7 相对地址寻址模式(RA)

在相对寻址模式中，指令的跳转范围只能在有符号数 -128 到 +127 之间。偏移量加上当前 PC 值，即为下一条要执行指令的地址。在加偏移量之前，PC 中的内容是紧跟着当前指令的后一条指令地址。

程序控制指令用相对寻址模式来实现条件跳转。支持相对寻址模式的指令有： BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE 和 JR 。

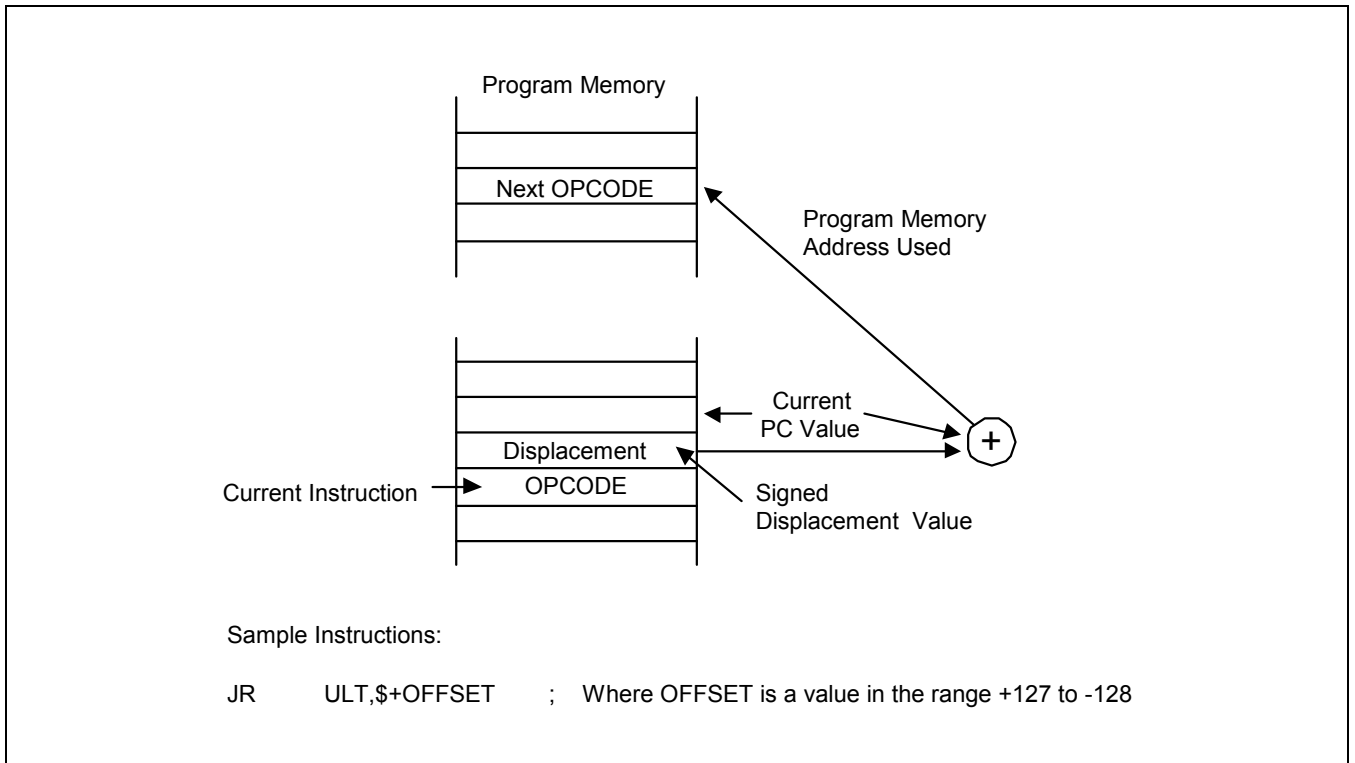


图 3-13 相对寻址

### 3.8 立即数寻址模式(IM)

立即数寻址模式中，操作数本身就包含在指令当中。根据指令的不同，操作数的长度可以是一个字节或一个字。立即数寻址模式常用来将常量赋值给寄存器。

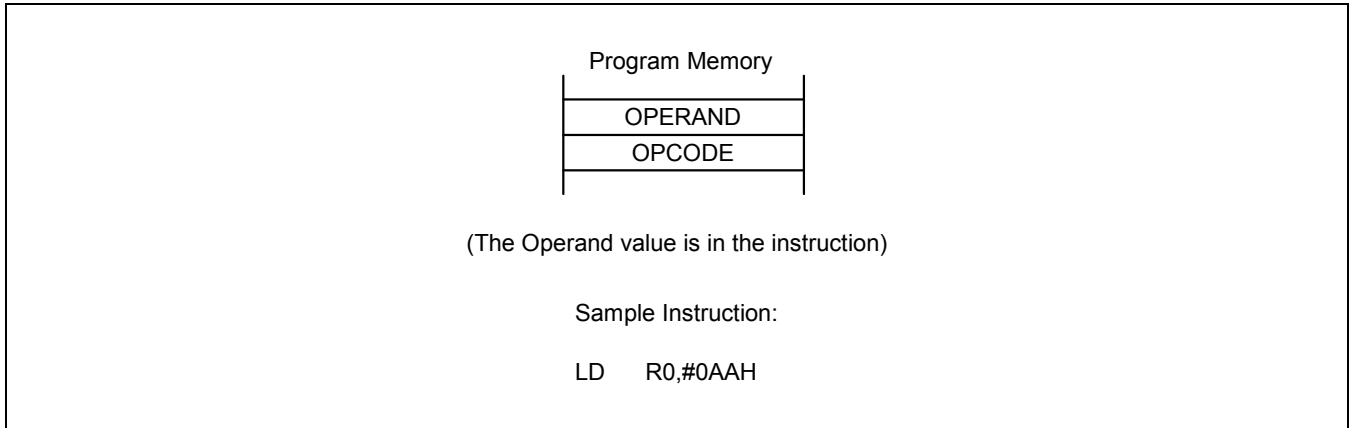


图 3-14 立即数寻址模式

# 4 控制寄存器

## 4.1 概述

本章中，S3C828BX/C8289X/C8285X控制寄存器的详细描述以易读方式排列。可以将本章视为编程时的快速参考资源。[表 4-1](#) 列出了标准寄存器描述的重要参数。

控制寄存器描述按照寄存器代表符号的字母顺序排列。更多有关控制寄存器的信息在本手册第二部分硬件资源描述中。

数据和计数寄存器没有在本章中详细列出。关于外设寄存器的更多内容详见本手册第二部分的相关外设章节。

[表 4-1](#) 列举了S3C828BX/C8289X/C8285X 所有映射寄存器的位置和读/写特性。每个映射寄存器的硬件复位值在第八章“复位和省电模式”中描述。

表 4-1 Set 1 寄存器

寄存器名	助记标号	十进制	十六进制	R/W
Basic Timer 控制寄存器	BTCON	211	D3H	R/W
系统时钟控制寄存器	CLKCON	212	D4H	R/W
系统标志寄存器	FLAGS	213	D5H	R/W
寄存器指针 0	RP0	214	D6H	R/W
寄存器指针 1	RP1	215	D7H	R/W
堆栈指针（高字节）	SPH	216	D8H	R/W
堆栈指针（低字节）	SPL	217	D9H	R/W
指令指针（高字节）	IPH	218	DAH	R/W
指令指针（低字节）	IPL	219	DBH	R/W
中断请求寄存器	IRQ	220	DCH	R
中断屏蔽寄存器	IMR	221	DDH	R/W
系统模式寄存器	SYM	222	DEH	R/W
寄存器页指针	PP	223	DFH	R/W

表 4-2 Set 1, Bank 0 寄存器

寄存器名	助记标号	十进制	十六进制	R/W
LCD 控制寄存器	LCON	208	D0H	R/W
钟表定时器控制寄存器	WTCON	209	D1H	R/W
电池电压检测控制寄存器	BLDCON	210	D2H	R/W
SIO 控制寄存器	SIOCON	224	E0H	R/W
SIO 数据寄存器	SIODATA	225	E1H	R/W
SIO 预处理寄存器	SIOPS	226	E2H	R/W
Timer 0 控制寄存器	T0CON	227	E3H	R/W
Timer 0 计数器(高字节)	T0CNTH	228	E4H	R
Timer 0 计数器(低字节)	T0CNTL	229	E5H	R
Timer 0 数据寄存器(高字节)	T0DATAH	230	E6H	R/W
Timer 0 数据寄存器(低字节)	T0DATAL	231	E7H	R/W
Timer A 控制寄存器	TACON	232	E8H	R/W
Timer A 计数器	TACNT	233	E9H	R
Timer A 数据寄存器	TADATA	234	EAH	R/W
Timer 1 控制寄存器	T1CON	235	EBH	R/W
Timer 1 计数器(高字节)	T1CNTH	236	ECH	R
Timer 1 计数器(低字节)	T1CNTL	237	EDH	R
Timer 1 数据寄存器(高字节)	T1DATAH	238	EEH	R/W
Timer 1 数据寄存器(低字节)	T1DATAL	239	EFH	R/W
Timer B 数据寄存器(高字节)	TBDATAH	240	F0H	R/W
Timer B 数据寄存器(低字节)	TBDAATAL	241	F1H	R/W
Timer B 控制寄存器	TBCON	242	F2H	R/W
A/D 转换控制寄存器	ADCON	243	F3H	R/W
A/D 转换数据寄存器(高字节)	ADDATAH	244	F4H	R
A/D 转换数据寄存器(低字节)	ADDAATAL	245	F5H	R
UART 控制寄存器	UARTCON	246	F6H	R/W
UART 数据寄存器	UDATA	247	F7H	R/W
UART 波特率数据寄存器	BRDATA	248	F8H	R/W
中断标志寄存器	INTPND	249	F9H	R/W
时钟控制寄存器	OSCCON	250	FAH	R/W
STOP 控制寄存器	STPCON	251	FBH	R/W
FCH 地址空间未映射				
Basic Timer 计数器	BTCNT	253	FDH	R
FEH 地址空间未映射				
中断优先级寄存器	IPR	255	FFH	R/W

表 4-3 Set 1, Bank 1 寄存器

寄存器名	助记标号	十进制	十六进制	R/W
闪存扇区地址寄存器(高字节)	FMSECH	208	D0H	R/W
闪存扇区地址寄存器(低字节)	FMSECL	209	D1H	R/W
闪存 控制寄存器	FMCON	210	D2H	R/W
P0 口控制寄存器 (高字节)	P0CONH	224	E0H	R/W
P0 口控制寄存器 (低字节)	P0CONL	225	E1H	R/W
P0 口中断控制寄存器 (高字节)	P0INTH	226	E2H	R/W
P0 口中断控制寄存器 (低 字节)	P0INTL	227	E3H	R/W
P0 口中断标志寄存器	P0PND	228	E4H	R/W
P1 口控制寄存器 (高字节)	P1CONH	229	E5H	R/W
P1 口控制寄存器 (低字节)	P1CONL	230	E6H	R/W
P1 口上拉电阻使能控制寄存器	P1PUR	231	E7H	R/W
P2 口控制寄存器 (高字节)	P2CONH	232	E8H	R/W
P2 口控制寄存器 (低字节)	P2CONL	233	E9H	R/W
P3 口控制寄存器 (高字节)	P3CONH	234	EAH	R/W
P3 口控制寄存器 (低字节)	P3CONL	235	EBH	R/W
P4 口控制寄存器 (高字节)	P4CONH	236	ECH	R/W
P4 口控制寄存器 (低字节)	P4CONL	237	EDH	R/W
P4 口上拉电阻使能控制寄存器	P4PUR	238	EEH	R/W
P5 口上拉电阻使能控制寄存器	P5PUR	239	EFH	R/W
P0 口数据寄存器	P0	240	F0H	R/W
P1 口数据寄存器	P1	241	F1H	R/W
P2 口数据寄存器	P2	242	F2H	R/W
P3 口数据寄存器	P3	243	F3H	R/W
P4 口数据寄存器	P4	244	F4H	R/W
P5 口数据寄存器	P5	245	F5H	R/W
P6 口数据寄存器	P6	246	F6H	R/W
P7 口数据寄存器	P7	247	F7H	R/W
P8 口数据寄存器	P8	248	F8H	R/W
P5 口控制寄存器 (高字节)	P5CONH	249	F9H	R/W
P5 口控制寄存器 (低字节)	P5CONL	250	FAH	R/W
P6 口控制寄存器 (高字节)	P6CONH	251	FBH	R/W
P6 口控制寄存器 (低字节)	P6CONL	252	FCH	R/W
P7 口控制寄存器	P7CON	253	FDH	R/W
P8 口控制寄存器	P8CON	254	FEH	R/W
闪存 用户可编程使能寄存器	FMUSR	255	FFH	R/W

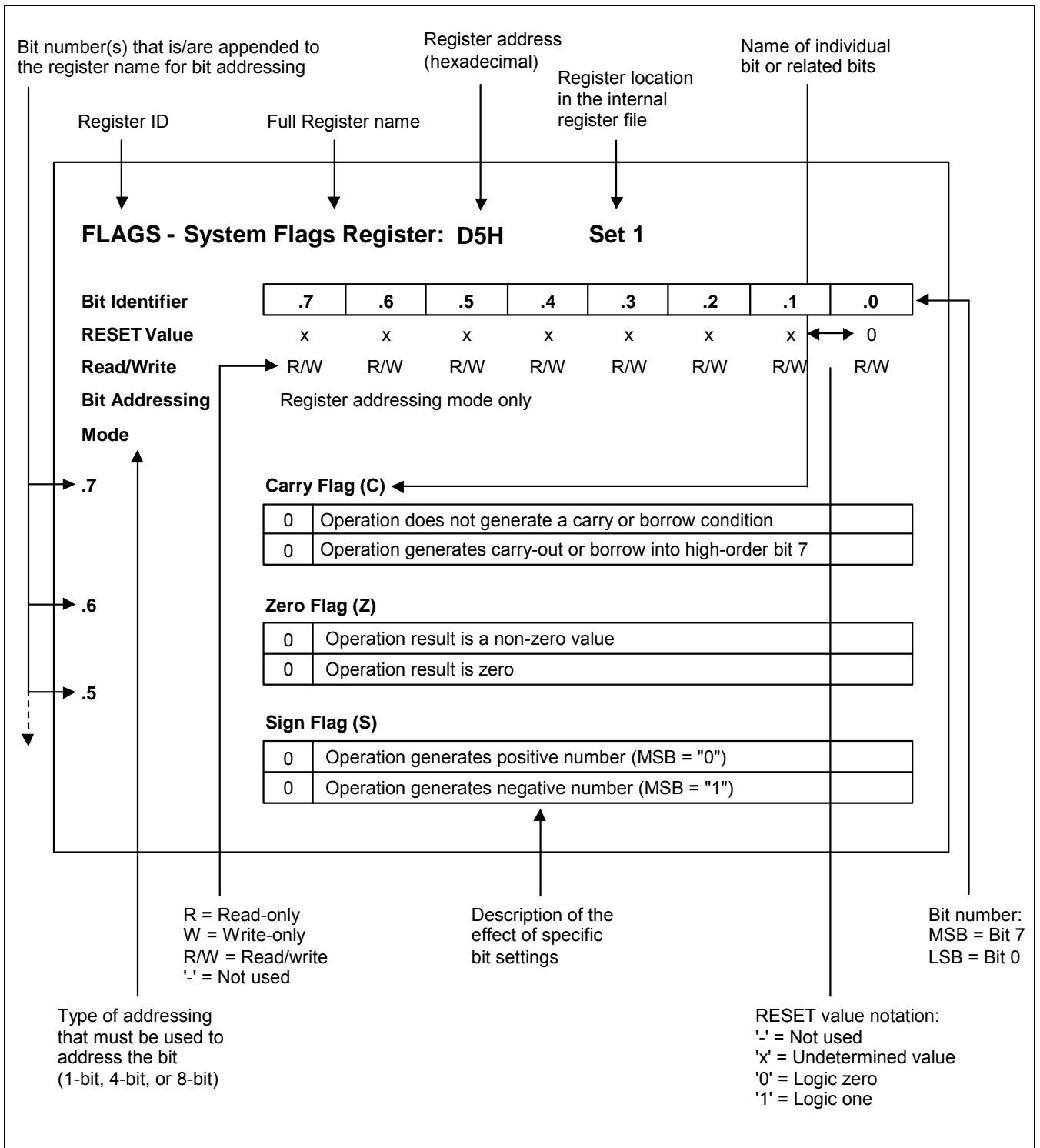


图 4-1 寄存器描述格式

## 4.1.1 ADCON — A/D 转换控制寄存器: F3H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	—	0	0	0	0	0	0	0
读/写	—	R/W	R/W	R	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7

S3C828BX/C8289X/C8285X 不使用

.6-.4

A/D输入管脚选择

0	0	0	AD0
0	0	1	AD1
0	1	0	AD2
0	1	1	AD3
1	0	0	AD4
1	0	1	AD5
1	1	0	AD6
1	1	1	AD7

.3

转换结束(EOC)状态位 (只读)

0	转换未完成
1	转换已完成

.2-.1

时钟源选择位

0	0	fxx/16
0	1	fxx/8
1	0	fxx/4
1	1	fxx/1

.0

A/D 转换启动位

0	禁止转换
1	启动转换



## 4.1.2 BLDCON — 电池电压检测控制寄存器: D2H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	—	—	0	0	0	0	0	0
读/写	—	—	R/W	R	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

S3C828BX/C8289X/C8285X 不使用

.5

VIN 源选择位

0	内部源
1	外部源

.4

BLD 检测结果输出位 (只读)

0	$V_{IN} > V_{REF}$ (使能BLD时)
1	$V_{IN} < V_{REF}$ (禁止BLD时)

.3

BLD 使能/禁止位

0	禁止BLD
1	使能BLD

.2-.0

检测电压阈值选择

0	0	0	$V_{BLD} = 2.2V$
1	0	1	$V_{BLD} = 2.4V$
0	1	1	$V_{BLD} = 2.8V$

## 4.1.3 BTCON — BASIC TIMER 控制寄存器: D3H - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.4

## 看门狗时钟功能使能位(系统复位)

1	0	1	0	禁止看门狗时钟功能
其它				使能看门狗时钟功能

.3-.2

Basic Timer 输入时钟选择位<sup>(3)</sup>

0	0	fx/4096
0	1	fx/1024
1	0	fx/128
1	1	fx/16

.1

Basic Timer 计数器清0控制位<sup>(1)</sup>

0	没有作用
1	清除basic timer 计数器值

.0

Basic Timer 和 Timer/Counters 分频器清0<sup>(2)</sup>

0	没有作用
1	清除2 者的计数器值

## 注释:

1. 当写“1”到 BTCON.1 时, Basic Timer 计数器被清 0, 之后该位也自动清 0。
2. 当写“1”到 BTCON.0 时, 相应的分频器被清 0, 之后该位也自动清 0。
3. fxx 为选定的系统时钟(主时钟或副时钟)。

## 4.1.4 CLKCON — 系统控制寄存器: D4H - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	—	—	0	0	—	—	—
读/写	R/W	—	—	R/W	R/W	—	—	—
寻址模式	仅寄存器寻址模式							

.7

振荡器 IRQ 唤醒功能位

0	使能省电模式下的 IRQ 唤醒主时钟
1	禁止省电模式下的 IRQ 唤醒主时钟

.6-.5

S3C828BX/C8289X/C8285X 不使用

.4-.3

CPU 时钟 (系统时钟) 选择位 (注)

0	0	fxx/16
0	1	fxx/8
1	0	fxx/2
1	1	fxx/1

.2-.0

S3C828BX/C8289X/C8285X 不使用

**注释:** 复位后, 选择最慢时钟 (16 分频) 作为系统时钟。可通过向 CLKCON.3 和 CLKCON.4 位写合适的值选择更快的时钟速率。

## 4.1.5 FLAGS — 系统标志寄存器: D5H - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
寻址模式	仅寄存器寻址模式							

.7

**Carry Flag (C)**

0	操作没有产生进位或借位
1	操作产生进位或者借位

.6

**Zero Flag (Z)**

0	操作结果不是“0”
1	操作结果是“0”

.5

**Sign Flag (S)**

0	操作产生正数 (MSB = “0”)
1	操作产生负数 (MSB = “1”)

.4

**Overflow Flag (V)**

0	操作结果在 -128 ~ + 127 之间
1	操作结果不在 -128 ~ + 127 之间, 即溢出

.3

**Decimal Adjust Flag (D)**

0	加操作完成
1	减操作完成

.2

**Half-Carry Flag (H)**

0	加法操作时第 3 位未产生进位或减法操作时第 3 位未产生借位
1	加法操作时第 3 位未产生进位或减法操作时第 3 位未产生借位

.1

**Fast Interrupt Status Flag (FIS)**

0	中断返回 (IRET) 正在进行 (读此位时)
1	快速中断服务程序正在进行 (读此位时)

.0

**Bank Address Selection Flag (BA)**

0	选择 Bank 0
1	选择 Bank 1

## 4.1.6 FMCON — 闪存 控制寄存器: D2H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	—	—	0
读/写	R/W	R/W	R/W	R/W	R	—	—	R/W
寻址模式	仅寄存器寻址模式							

.7-.4

## 闪存模式选择位

0	1	0	1	(字节) 编程模式
1	0	1	0	扇区擦除模式
0	1	1	0	Hard Lock 模式
其他				不可用

.3

## 扇区擦除状态位 (只读)

0	扇区擦除成功
1	扇区擦除失败

.2-.1

S3C828BX/C8289X/C8285X不使用

.0

## 闪存操作启动位

0	操作停止
1	操作开始

注释: FMCON.0 在相应的操作完成后会自动清 0.

## 4.1.7 FMSECH — 闪存扇区地址寄存器 (高字节): D0H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7-0** 闪存 扇区地址位 (高字节)

闪存扇区选择高8位 (第15 ~ 8位)。
-----------------------

注释: 高字节 闪存 扇区地址指针指向 16 位指针地址的高 8 位。

## 4.1.8 FMSECL — 闪存扇区地址寄存器 (低字节): D1H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
寻址模式	仅寄存器寻址模式							

**.7** 闪存 扇区地址位 (低字节)

闪存 扇区选择第 7 位。
---------------

**.6-0** S3C828BX/C8289X/C8285X不使用

注释: 低字节 闪存 扇区地址指针指向 16 位指针地址的低 8 位。

## 4.1.9 FMUSR — 闪存用户可编程使能寄存器: FFH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.0

闪存用户可编程使能位

1	0	1	0	0	1	0	1	使能用户编程模式
其它								禁止用户编程模式

## 4.1.10 IMR — 中断屏蔽寄存器: DDH - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7 中断优先级7 (IRQ7) 使能位; 外部中断 P0.4–0.7, P3.0–3.3**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

**.6 中断优先级6 (IRQ6) 使能位; 外部中断 P0.0–0.3**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

**.5 中断优先级5 (IRQ5) 使能位; Frequency Counter 溢出, Frequency Counter 门周期, Comparator 0/1**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

**.4 中断优先级4 (IRQ4) 使能位; SIO**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

**.3 中断优先级3 (IRQ3) 使能位; UART 0/1 发送, UART 0/1 接收**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

**.2 中断优先级2 (IRQ2) 使能位; Timer A 匹配/捕获, Timer A 溢出, 钟表定时器**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

**.1 中断优先级1 (IRQ1) 使能位; Timer B 匹配**

0	禁止 (屏蔽)
1	使能 (未屏蔽)



.0

中断优先级0 (IRQ0) 使能位; Timer 0 匹配/捕获, Timer 0 溢出, Timer 1 匹配/捕获, Timer 1 溢出

0	禁止 (屏蔽)
1	使能 (未屏蔽)

**注释:** 当某个中断级被屏蔽, 任何发起的中断请求都不能被 CPU 识别。

## 4.1.11 INTPND — 中断标志寄存器: F9H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

S3C828BX/C8289X/C8285X 不使用

.5

**Rx 中断标志位(UART)**

0	无中断请求(读此位时), 写入0清除中断标志
1	中断请求置起(读此位时)

.4

**Tx 中断标志位(UART)**

0	无中断请求(读此位时), 写入0清除中断标志
1	中断请求置起(读此位时)

.3

**Timer 1 匹配/捕获中断标志位**

0	无中断请求(读此位时), 写入0清除中断标志
1	中断请求置起(读此位时)

.2

**Timer 1 溢出中断标志位**

0	无中断请求(读此位时), 写入0清除中断标志
1	中断请求置起(读此位时)

.1

**Timer A匹配/捕获中断标志位**

0	无中断请求(读此位时), 写入0清除中断标志
1	中断请求置起(读此位时)

.0

**Timer A溢出中断标志位**

0	无中断请求(读此位时), 写入0清除中断标志
1	中断请求置起(读此位时)

## 4.1.12 IPH — 指令指针 (HIGH BYTE): DAH - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-0

## 指令指针地址（高字节）

高字节指令指针指向 16 位指令指针地址的高 8 位（IP15-IP8）。  
IP 地址的低字节在 IPL 寄存器（地址：DBH）中。

## 4.1.13 IPL — 指令指针(低字节): DBH - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-0

## 指令指针地址（低字节）

低字节指令指针指向 16 位指令指针地址的低 8 位（IP7-IP0）。  
IP 地址的高字节在 IPH 寄存器（地址：DAH）中。

## 4.1.14 IPR — 中断优先级寄存器: FFH - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7, .4, and .1**

## 中断 A、B 和 C 组优先级控制位

0	0	0	组优先级未定义
0	0	1	B > C > A
0	1	0	A > B > C
0	1	1	B > A > C
1	0	0	C > A > B
1	0	1	C > B > A
1	1	0	A > C > B
1	1	1	组优先级未定义

**.6**

## 中断 C 子分组优先级控制位

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

**.5**

## 中断 C 组优先级控制位

0	IRQ5 > (IRQ6, IRQ7)
1	(IRQ6, IRQ7) > IRQ5

**.3**

## 中断 B 子分组优先级控制位

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

**.2**

## 中断 B 组优先级控制位

0	IRQ2 > (IRQ3, IRQ4)
1	(IRQ3, IRQ4) > IRQ2

**.0**

## 中断 A 组优先级控制位

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

注释: 中断 A 组 - IRQ0, IRQ1  
 中断 B 组 - IRQ2, IRQ3, IRQ4  
 中断 C 组 - IRQ5, IRQ6, IRQ7

## 4.1.15 IRQ —中断请求寄存器: DCH - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R	R	R	R	R	R	R	R
寻址模式	仅寄存器寻址模式							

**.7 中断级 7 (IRQ7) 请求标志位; 外部中断 P0.4–0.7, P3.0–3.3**

0	没有中断
1	标志位置位

**.6 中断级 6 (IRQ6) 请求标志位; 外部中断 P0.0–0.3**

0	没有中断
1	标志位置位

**.5 中断级 5 (IRQ5) 请求标志位; Frequency Counter 溢出, Frequency Counter 门周期, Comparator 0/1**

0	没有中断
1	标志位置位

**.4 中断级 4 (IRQ4) 请求标志位; SIO**

0	没有中断
1	标志位置位

**.3 中断级 3 (IRQ3) 请求标志位; UART 0/1 发送, UART 0/1 接收**

0	没有中断
1	标志位置位

**.2 中断级 2 (IRQ2) 请求标志位; Timer A 匹配/捕获, Timer A 溢出, 钟表定时器**

0	没有中断
1	标志位置位

**.1 中断级 1 (IRQ1) 请求标志位; Timer B 匹配**

0	没有中断
1	标志位置位

.0

中断级 0 (IRQ0) 请求标志位; Timer 0 匹配/捕获, Timer 0 溢出, Timer 1 匹配/捕获, Timer 1 溢出

0	没有中断
1	标志位置位

## 4.1.16 LCON — LCD 控制寄存器: D0H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	~	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	~	R/W
寻址模式	仅寄存器寻址模式							

**.7** 内部 LCD 分压电阻使能位

0	使能LCD内部分压电阻
1	禁止LCD内部分压电阻

**.6–5** LCD 时钟源选择位

0	0	$f_w/2^8$ (128Hz)
0	1	$f_w/2^7$ (256Hz)
1	0	$f_w/2^6$ (512Hz)
1	1	$f_w/2^5$ (1024Hz)

**.4–2** LCD 占空比和偏执选择位 (注释)

0	0	0	1/8占空比, 1/4 偏置
0	0	1	1/4占空比, 1/3 偏置
0	1	0	1/3占空比, 1/3 偏置
0	1	1	1/3占空比, 1/2 偏置
1	x	x	1/2占空比, 1/2 偏置

**.1** S3C828BX/C8289X/C8285X 不使用**.0** LCD 显示控制位

0	所有 LCD 信号为低电平(关闭P-Tr)
1	打开显示 (打开 P-Tr)

## 注释:

- "x" 表示值可以为任意值.
- 当选择 1/3 偏置时, 偏置电压可设为  $V_{LC0}$ ,  $V_{LC1}$ ,  $V_{LC2}$  ( $V_{LC3}$ ), 和  $V_{SS}$ .
- 当选择 1/2 偏置时, 偏置电压可设为  $V_{LC0}$ ,  $V_{LC1}$  ( $V_{LC2}$ ,  $V_{LC3}$ ), 和  $V_{SS}$ .

## 4.1.17 OSCCON — 时钟 控制寄存器: FAH - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	—	—	—	0	0	—	0
读/写	R/W	—	—	—	R/W	R/W	—	R/W
寻址模式	仅寄存器寻址模式							

.7

## 副时钟电路选择位

0	初始状态
1	副振荡器省电模式选择 (当副振荡器停止时自动清 "0")

.6-.4

S3C828BX/C8289X/C8285X 不使用

.3

## 主振荡器控制位

0	主振荡器工作
1	主振荡器停止

.2

## 副振荡器控制位

0	副振荡器工作
1	副振荡器停止

.1

S3C828BX/C8289X/C8285X 不使用

.0

## 系统时钟选择位

0	使用主振荡器作系统时钟
1	使用副振荡器作系统时钟

## 注释:

1. 在 VREG 和 GND 之间需要加一个电容 (0.1uF)。
2. 副振荡器工作时 OSCCON.7 必须保持 "1"。



## 4.1.18 P0CONH — P0 控制寄存器 (高字节): E0H - SET 1, BANK 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P0.7/INT7**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

.5-4

**P0.6/INT6**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

.3-2

**P0.5/INT5**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

.1-0

**P0.4/INT4**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

## 4.1.19 P0CONL — P0 口控制寄存器 (低字节): E1H - SET 1, BANK 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

**P0.3/INT3**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

.5-.4

**P0.2/INT2**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

.3-.2

**P0.1/INT1**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

.1-.0

**P0.0/INT0**

0	0	施密特输入模式
0	1	施密特输入模式, 带上拉电阻
1	0	开漏输出
1	1	推挽输出

## 4.1.20 P0INTH — P0 口中断控制寄存器 (高字节): E2H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

**P0.7/外部中断 (INT7) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

.5-.4

**P0.6/外部中断 (INT6) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

.3-.2

**P0.5/外部中断 (INT5) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

.1-.0

**P0.4/外部中断 (INT4) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

## 4.1.21 P0INTL — P0 中断控制寄存器 (低字节): E3H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

**P0.3/外部中断 (INT3) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

.5-.4

**P0.2/外部中断 (INT2) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

.3-.2

**P0.1/外部中断 (INT1) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

.1-.0

**P0.0/外部中断 (INT0) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	使能下降沿和上升沿中断

## 4.1.22 P0PND — P0 中断标志寄存器: E4H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7 P0.7/外部中断(INT7) 标志位**

0	清除中断(写此位时)
1	P0.7/INT7 中断请求置位(读此位时)

**.6 P0.6/外部中断(INT6) 标志位**

0	清除中断(写此位时)
1	P0.6/INT6 中断请求置位(读此位时)

**.5 P0.5/外部中断(INT5) 标志位**

0	清除中断(写此位时)
1	P0.5/INT5 中断请求置位(读此位时)

**.4 P0.4/外部中断(INT4) 标志位**

0	清除中断(写此位时)
1	P0.4/INT4 中断请求置位(读此位时)

**.3 P0.3/外部中断(INT3) 标志位**

0	清除中断(写此位时)
1	P0.3/INT3 中断请求置位(读此位时)

**.2 P0.2/外部中断(INT2) 标志位**

0	清除中断(写此位时)
1	P0.2/INT2 中断请求置位(读此位时)

**.1 P0.1/外部中断(INT1) 标志位**

0	清除中断(写此位时)
1	P0.1/INT1 中断请求置位(读此位时)

**.0 P0.0/外部中断(INT0) 标志位**

0	清除中断(写此位时)
1	P0.0/INT0 中断请求置位(读此位时)

## 4.1.23 P1CONH — P1 控制寄存器 (高字节): E5H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

S3C828BX/C8289X/C8285X不使用

.5-.4

## P1.6/SI

0	0	施密特输入模式(SI)
0	1	N 沟道开漏输出模式
1	0	推挽输出模式
1	1	S3C828BX/C8289X/C8285X不使用

.3-.2

## P1.5/SCK

0	0	施密特输入模式(SCK 输入)
0	1	N 沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SCK 输出)

.1-.0

## P1.4/SO

0	0	施密特输入模式
0	1	N 沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能(SO)

## 4.1.24 P1CONL — P1 口控制寄存器 (低字节): E6H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P1.3/BUZ**

0	0	施密特输入模式
0	1	N 沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能(BUZ)

.5-4

**P1.2/T1OUT/T1PWM**

0	0	施密特输入模式
0	1	N 沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能(T1OUT/T1PWM)

.3-2

**P1.1/T1CLK**

0	0	施密特输入模式 (T1CLK)
0	1	N 沟道开漏输出模式
1	0	推挽输出模式
1	1	S3C828BX/C8289X/C8285X 不使用

.1-0

**P1.0/T1CAP**

0	0	施密特输入模式 (T1CAP)
0	1	N 沟道开漏输出模式
1	0	推挽输出模式
1	1	S3C828BX/C8289X/C8285X 不使用

## 4.1.25 P1PUR — P1 口上拉电阻使能控制寄存器: E7H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7** **S3C828BX/C8289X/C8285X 不使用**

**.6** **P1.6 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.5** **P1.5 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.4** **P1.4 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.3** **P1.3 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.2** **P1.2 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.1** **P1.1 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.0** **P1.0 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**注释:** 当相应的口设为推挽输出或复用功能时, 其上拉电阻自动禁止。



## 4.1.26 P2CONH — P2 控制寄存器 (高字节): E8H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

**P2.7/AD7/VBLDREF**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能 (AD7/VBLDREF)

.5-.4

**P2.6/AD6**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能(AD6)

.3-.2

**P2.5/AD5**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能(AD5)

.1-.0

**P2.4/AD4**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能(AD4)

## 4.1.27 P2CONL — P2 口控制寄存器 (低字节): E9H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P2.3/AD3**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能(AD3)

.5-4

**P2.2/AD2**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能(AD2)

.3-2

**P2.1/AD1**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能(AD1)

.1-0

**P2.0/AD0**

0	0	输入模式
0	1	输入带上拉电阻
1	0	推挽输出
1	1	复用功能(AD0)

## 4.1.28 P3CONH — P3 口控制寄存器 (高字节): EAH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	–	–	0	0	0	0	0	0
读/写	–	–	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7–.6

S3C828BX/C8289X/C8285X 不使用

.5

P3.1/TAOUT/TAPWM/SEG35 (只在P3CONL.3-.2 = "11" 时)

0	TAOUT/TAPWM输出
1	SEG35 输出

.4

P3.0/TBPWM/SEG34 (只在P3CONL.3-.2 = "11" 时)

0	TBPWM 输出
1	SEG34 输出

.3–.2

P3.5/RxD

0	0	输入模式 (RxD)
0	1	输入带上拉(RxD)
1	0	推挽输出
1	1	复用功能 (RxD out)

.1–.0

P3.4/TxD

0	0	输入模式
0	1	输入带上拉
1	0	推挽输出
1	1	复用功能(TxD)

## 4.1.29 P3CONL — P3 口控制寄存器 (低字节): EBH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

**P3.3/TACAP/SEG37**

0	0	输入模式(TACAP)
0	1	输入带上拉(TACAP)
1	0	推挽输出
1	1	复用功能(SEG37)

.5-.4

**P3.2/TACLK/SEG36**

0	0	输入模式(TACLK)
0	1	输入带上拉(TACLK)
1	0	推挽输出
1	1	复用功能(SEG36)

.3-.2

**P3.1/TAOUT/TAPWM/SEG35**

0	0	输入模式
0	1	输入带上拉
1	0	推挽输出
1	1	复用功能(TAOUT/TAPWM/SEG35)

.1-.0

**P3.0/TBPWM/SEG34**

0	0	输入模式
0	1	输入带上拉
1	0	推挽输出
1	1	复用功能(TBPWM/SEG34)

## 4.1.30 P4CONH — P4 控制寄存器 (高字节): ECH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P4.7/SEG25**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能 (SEG25)

.5-4

**P4.6/SEG24**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG24)

.3-2

**P4.5/SEG23**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG23)

.1-0

**P4.4/SEG22**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG22)

## 4.1.31 P4CONL — P4 口控制寄存器 (低字节): EDH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

## P4.3/SEG21

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG21)

.5-4

## P4.2/SEG20

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG20)

.3-2

## P4.1/SEG19

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG19)

.1-0

## P4.0/SEG18

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG18)

## 4.1.32 P4PUR — P4 口上拉电阻使能位: EEH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7 P4.7 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.6 P4.6 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.5 P4.5 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.4 P4.4 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.3 P4.3 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.2 P4.2 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**.1 P4.1 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.0

**P4.0上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**注释:** 当相应的口设为推挽输出或复用功能时，其上拉电阻自动禁止。



## 4.1.33 P5CONH — P5 口控制寄存器 (高字节): F9H - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P5.7/SEG33**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG33)

.5-4

**P5.6/SEG32**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG32)

.3-2

**P5.5/SEG31**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG31)

.1-0

**P5.4/SEG30**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG30)

## 4.1.34 P5CONL — P5 口控制寄存器 (低字节): FAH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P5.3/SEG29**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG29)

.5-4

**P5.2/SEG28**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG28)

.3-2

**P5.1/SEG27**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG27)

.1-0

**P5.0/SEG26**

0	0	输入模式
0	1	N-沟道开漏输出
1	0	推挽输出
1	1	复用功能(SEG26)

## 4.1.35 P5PUR — P5 口上拉电阻使能控制寄存器: EFH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7

**P5.7上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.6

**P5.6上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.5

**P5.5上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.4

**P5.4上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.3

**P5.3上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.2

**P5.2上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.1

**P5.1上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.0

**P5.0上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

**注释:** 当相应的口设为推挽输出或复用功能时，其上拉电阻自动禁止。

## 4.1.36 P6CONH — P6 口控制寄存器 (高字节): FBH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P6.7/SEG17**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能 (SEG17)

.5-4

**P6.6/SEG16**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG16)

.3-2

**P6.5/SEG15**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG15)

.1-0

**P6.4/SEG14**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG14)

## 4.1.37 P6CONL — P6 口控制寄存器(低字节): FCH- SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P6.3/SEG13**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG13)

.5-4

**P6.2/SEG12**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG12)

.3-2

**P6.1/SEG11**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG11)

.1-0

**P6.0/SEG10**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG10)

## 4.1.38 P7CON — P7 口控制寄存器: FDH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-6

**P7.3/SEG9**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG9)

.5-4

**P7.2/SEG8**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG8)

.3-2

**P7.1/SEG7**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG7)

.1-0

**P7.0/SEG6**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(SEG6)

## 4.1.39 P8CON — P8 口控制寄存器: FEH - SET 1, BANK1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

**P8.7-P8.4/COM7-COM4/SEG5-SEG2**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(COM7-COM4/SEG5-SEG2)

.5-.4

**P8.3/COM3/SEG1**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(COM3/SEG1)

.3-.2

**P8.2/COM2/SEG0**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(COM2/SEG0)

.1-.0

**P8.1-P8.0/COM1-COM0**

0	0	输入模式
0	1	输入上拉模式
1	0	推挽输出模式
1	1	复用功能(COM1-COM0)



## 4.1.40 PP — 寄存器页指针: DFH - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

## .7-.4

## 目的寄存器页选择为

0	0	0	0	目的寄存器页: 第0 页
0	0	0	1	目的寄存器页: 第1 页
0	0	1	0	目的寄存器页: 第2 页(S3C8285X 不使用)
0	0	1	1	目的寄存器页: 第3 页(S3C8285X不使用)
0	1	0	0	目的寄存器页: 第4 页(S3C8289X/C8285X不使用)
0	1	0	1	目的寄存器页: 第5 页(S3C8289X/C8285X不使用)
0	1	1	0	目的寄存器页: 第6 页(S3C8289X/C8285X不使用)
0	1	1	1	目的寄存器页: 第7 页(S3C8289X/C8285X不使用)
1	0	0	0	目的寄存器页: 第8 页(S3C8289X/C8285X不使用)
1	0	0	1	目的寄存器页: 第9 页(S3C8289X/C8285X不使用)
1	1	1	1	目的寄存器页: 第15 页
其它				S3C828BX/C8289X/C8285X不使用

## .3-.0

## 源寄存器页选择位

0	0	0	0	源寄存器页: 第 0 页
0	0	0	1	源寄存器页: 第 1 页
0	0	1	0	源寄存器页: 第 2 页(S3C8285X不使用)
0	0	1	1	源寄存器页: 第 3 页(S3C8285X不使用)
0	1	0	0	源寄存器页: 第 4 页(S3C8289X/C8285X不使用)
0	1	0	1	源寄存器页: 第 5 页 (S3C8289X/C8285X不使用)
0	1	1	0	源寄存器页: 第 6 页 (S3C8289X/C8285X不使用)
0	1	1	1	源寄存器页: 第 7 页 (S3C8289X/C8285X不使用)
1	0	0	0	源寄存器页: 第 8 页(S3C8289X/C8285X不使用)
1	0	0	1	源寄存器页: 第 9 页(S3C8289X/C8285X不使用)
1	1	1	1	源寄存器页: 第 15 页
其它				S3C828BX/C8289X/C8285X不使用

**注释:**

1. S3C828BX 内部寄存器卷分为 11 页 (页 0-9, 15)..  
页 0-9 作为通用寄存器卷使用
2. S3C8289X 内部寄存器卷分为 5 页 (页 0-3, 15).  
页 0-3 作为通用寄存器卷使用
3. S3C8285X 内部寄存器卷分为 3 页 (页 0-1, 15).  
页 0-1 作为通用寄存器卷使用
4. 页 15 作为 LCD 数据寄存器和通用寄存器使用

## 4.1.41 RP0 — 寄存器指针 0: D6H - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	1	1	0	0	0	—	—	—
读/写	R/W	R/W	R/W	R/W	R/W	—	—	—
寻址模式	仅寄存器寻址模式							

.7-.3

**寄存器指针 0 地址值**

寄存器指针 0 可以单独指向寄存器卷中的某个 256 字节工作寄存器区域。通过使用寄存器指针 RP0 和 RP1 同时选择 2 个 8 字节寄存器组作为有效的工作寄存器空间。复位后，RP0 指向寄存器 Set 1 的 C0H 地址，选择从 C0H 到 C7H 的 8 位工作寄存器。

.2-.0

**S3C828BX/C8289X/C8285X 不使用**

## 4.1.42 RP1 — 寄存器指针 1: D7H - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	1	1	0	0	1	—	—	—
读/写	R/W	R/W	R/W	R/W	R/W	—	—	—
寻址模式	仅寄存器寻址模式							

.7-.3

**寄存器指针 1 地址值**

寄存器指针 1 可以单独指向寄存器卷中的某个 256 字节工作寄存器区域。通过使用寄存器指针 RP0 和 RP1 同时选择 2 个 8 字节寄存器组作为有效的工作寄存器空间。复位后，RP1 指向寄存器 Set 1 的 C8H 地址，选择从 C8H 到 CFH 的 8 位工作寄存器。

.2-.0

**S3C828BX/C8289X/C8285X 不使用**

## 4.1.43 SIOCON — SIO 控制寄存器: E0H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7 SIO 移位时钟选择位**

0	内部时钟 (P.S 时钟)
1	外部时钟 (SCK)

**.6 数据方向控制为**

0	MSB优先模式
1	LSB优先模式

**.5 SIO 模式选择位**

0	只接收模式
1	发送/接收模式

**.4 移位时钟边沿选择位**

0	下降沿 Tx, 上升沿 Rx
1	上升沿 Tx, 下降沿Rx

**.3 SIO 计数器清 0 和移位启动位**

0	无效
1	3位计数器清0, 并启动移位

**.2 SIO 移位操作时能位**

0	禁止移位和时钟计数
1	使能移位和时钟计数

**.1 SIO 中断使能位**

0	禁止 SIO 中断
1	使能 SIO 中断

**.0 SIO 中断标志位**

0	无中断标志 (读此位时), 清除中断标志 ()
1	中断标志位置位 (读此位时)

## 4.1.44 SPH — 堆栈指针 (高字节): D8H - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-0

## 堆栈指针地址 (高字节)

高字节堆栈指针的值是 16 位堆栈指针地址 (SP15-SP8) 的高 8 位。  
低字节堆栈指针的值位于寄存器 SPL (D9H) 中。  
复位后, 堆栈指针 (SP) 的值不确定。

## 4.1.45 SPL — 堆栈指针 (低字节): D9H - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-0

## 堆栈指针地址(低字节)

低字节堆栈指针的值是 16 位堆栈指针地址 (SP7-SP0) 的低 8 位。  
高字节堆栈指针的值位于寄存器 SPH (D8H) 中。  
复位后, 堆栈指针 (SP) 的值不确定。

## 4.1.46 STPCON — STOP 控制寄存器: FBH- SET 1, BANK 0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.0

## STOP 控制位

1 0 1 0 0 1 0 1	使能 使用 stop 指令
其它值	禁止 使用 stop 指令

**注释:** 在执行 STOP 指令前, 需要将 STPCON 寄存器设置为“10100101B”, 否则, STOP 指令不会被执行, 同时系统会复位

## 4.1.47 SYM — 系统模式寄存器: DEH - SET 1

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	–	–	x	x	x	0	0
读/写	R/W	–	–	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7** 始终保持逻辑 "0"

**.6–.5** S3C828BX/C8289X/C8285X不使用

**.4–.2** 快速中断优先级选择位 <sup>(1)</sup>

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

**.1** 快速中断使能位 <sup>(2)</sup>

0	禁止快速中断
1	使能快速中断

**.0** 全局中断使能位 <sup>(3)</sup>

0	禁止所有中断
1	使能所有中断

## 注释:

- 一次只能选择一个中断优先级作为快速中断。
- 将 SYM.1 设为 "1" 使能 SYM.2-SYM.4 选择的快速中断。
- 复位后, 必须通过执行 EI 指令使能全局中断 (而不是往 SYM.0 写 "1")。

## 4.1.48 T0CON — TIMER 0 控制寄存器: E3H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.5

## Timer 0 输入时钟选择位

0	0	0	TBOF
0	0	1	fx/256
0	1	0	fx/64
0	1	1	fx/8
1	x	x	fx/1

.4

S3C828BX/C8289X/C8285X不使用

.3

## Timer 0 计数器清0位

0	无效
1	清除 timer 0 计数器 (写此位时)

.2

## Timer 0 计数器使能位

0	禁止计数器计数
1	使能计数器计数

.1

## Timer 0 匹配中断使能位

0	禁止中断
1	使能中断

.0

## Timer 0 中断标志位

0	无中断标志 (读此位时), 清除中断标志 (写此位时)
1	T0 中断置位 (读此位时)

注释: T0CON.3 在计数器清 0 后自动清 0。



## 4.1.49 T1CON — TIMER 1 控制寄存器: EBH - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-5

## Timer 1 时钟源选择位

0	0	0	fx/1024
0	0	1	fx/256
0	1	0	fx/64
0	1	1	fx/8
1	0	0	fx/1
1	0	1	外部时钟 (T1CLK) 下降沿
1	1	0	外部时钟(T1CLK) 上升沿
1	1	1	计数其停止

.4-3

## Timer 1 运行模式选择位

0	0	定时模式 (T1OUT)
0	1	捕获模式 (上升沿捕获, 计数器计数, 可以产生溢出)
1	0	捕获模式 (下降沿沿捕获, 计数器计数, 可以产生溢出)
1	1	PWM 模式(可以产生溢出和匹配中断)

.2

## Timer 1 计数器清0位

0	没有作用
1	清除 timer 1 计数器(写此位时)

.1

## Timer 1 匹配/捕获中断使能位

0	禁止中断
1	使能中断

.0

## Timer 1 溢出中断使能位

0	禁止溢出中断
1	使能溢出中断

注释: T1CON.2 在计数器清 0 后自动清 0.

## 4.1.50 TACON — TIMER A 控制寄存器: E8H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.5

## Timer A 输入时钟源选择位

0	0	0	fx/1024
0	0	1	fx/256
0	1	0	fx/64
0	1	1	fx/8
1	0	0	fx/1(系统时钟)
1	0	1	外部时钟 (TACLK) 下降沿
1	1	0	外部时钟 (TACLK) 上升沿
1	1	1	计数器停止

.4-.3

## Timer A 运行模式选择位

0	0	定时模式(TAOUT)
0	1	捕获模式 (上升沿捕获, 计数器计数, 可以产生溢出)
1	0	捕获模式 (下降沿沿捕获, 计数器计数, 可以产生溢出)
1	1	PWM 模式(可以产生溢出中断)

.2

## Timer A 计数器清0 位

0	没有作用
1	清除 timer A 计数器(写此位时)

.1

## Timer A 匹配/捕获中断使能位

0	禁止中断
1	使能中断

.0

## Timer A 溢出中断使能位

0	禁止溢出中断
1	使能溢出中断

注释: TACON.2 在计数器清 0 后自动清 0.

## 4.1.51 TBCON — TIMER B 控制寄存器: F2H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

## Timer B 输入时钟选择位

0	0	fxx/1
0	1	fxx/2
1	0	fxx/4
1	1	fxx/8

.5-.4

## Timer B 中断时间点选择

0	0	低位数据借位后产生中断.
0	1	高位数据借位后产生中断
1	0	低位和高位数据数据借位后产生中断
1	1	不可用

.3

## Timer B 中断使能位

0	禁止中断
1	使能中断

.2

## Timer B 停止/启动位

0	停止 timer B
1	启动 timer B

.1

## Timer B 模式选择位

0	单脉冲模式
1	重复模式

.0

## Timer B 输出电平控制位

0	TBOF 为低 (TBPWM: 低位数据为低电平, 高位数据为高电平)
1	TBOF 为高 (TBPWM: 高位数据为低电平, 低位数据为高电平)

## 4.1.52 UARTCON — UART 控制寄存器: F6H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7-.6

## UART 模式选择位 Mode Selection Bits

0	0	Mode 0 模式: 移位寄存器 ( $fx/(16 \times (BRDATA+1))$ )
0	1	Mode 1 模式: 8位 UART ( $fx/(16 \times (BRDATA+1))$ )
1	0	Mode 2 模式: 9位 UART ( $fx/16$ )
1	1	Mode 3 模式: 9位 UART ( $fx/(16 \times (BRDATA+1))$ )

.5

## 多点通讯使能位 (只在 modes 2 和 3 模式下)

0	禁止
1	使能

.4

## 串行数据接收使能位

0	禁止
1	使能

.3

## TB8

在UART的 mode 2 或 3模式下要发送的第9位数据("0" or "1")

.2

## RB8

在UART的 mode 2 或 3模式下接收到的第9位数据("0" or "1")

.1

## 接收中断使能位

0	禁止Rx 中断
1	使能Rx 中断

.0

## 发送中断使能位

0	禁止Tx中断
1	使能Tx中断

**注释:**

1. 在 mode 2 或者 mode 3 模式下, 如果设置 UARTCON.5 位的值为"1", 则当收到的第 9 位数据位为"0"时, 不会产生接收中断。在 mode 1 模式下, 如果设置 UARTCON.5 位的值为"1", 则在没有收到有效的停止位的时候, 不会产出接收中断。  
在 mode 0 模式下, 应将 UARTCON.5 设为"0"
2. 有关 UART 的 8-bit 和 9-bit 的描述中, 其中的 8-bit 和 9-bit 是指有效数据的位数, 不包括发送和接收时的开始位和停止位。
3. 接收和发送中断标志位位于 INTPND 寄存器

## 4.1.53 WTCN — 钟表定时器控制寄存器: D1H - SET 1, BANK0

位定义	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	仅寄存器寻址模式							

**.7** 钟表定时器 时钟选择位

0	主时钟 27 分频(fxx/128)
1	副时钟(fxt)

**.6** 钟表定时器 中断时能位

0	禁止钟表定时器 中断
1	使能钟表定时器 中断

**.5-4** 蜂鸣器信号选择位

0	0	0.5kHz
0	1	1kHz
1	0	2kHz
1	1	4kHz

**.3-2** 钟表定时器 速度选择位

0	0	钟表定时器中断时间: 1.0s
0	1	钟表定时器中断时间: 0.5s
1	0	钟表定时器中断时间: 0.25s
1	1	钟表定时器中断时间: 3.91ms

**.1** 钟表定时器使能位

0	禁止钟表定时器; 分频电路清0
1	使能钟表定时器

**.0** 钟表定时器中断标志位

0	无中断标志 (读此位时), 清除中断标志 (写此位时)
1	中断标志位置位 (读此位时)

注释: 假设 钟表定时器 的时钟频率 (fw) 为 32.768kHz.

# 5

## 中断结构

### 5.1 概述

S3C8- 系列的中断结构由三个基本部分组成：中断级，中断向量和中断源。SAM8 CPU 最多可以识别 8 个中断级和 128 个中断向量。当多个中断向量同属于一个中断级时，这几个向量的优先级由硬件决定。从芯片设计的角度来看，单个或多个中断源可以共用同一个中断向量地址。

#### 5.1.1 中断级 (LEVELS)

中断级是优先级分配和识别的主要单元。所有的外设和 I/O 模块都可以发出中断请求。换句话说，外设和 I/O 模块的操作都是中断驱动的 (interrupt-driven)。一共有 8 个可能的中断级：IRQ0-IRQ7，也被称为优先级 0~优先级 7。每个中断级直接与中断请求号 (IRQn) 关联。每款芯片的中断级数量不同。

S3C828BX/C8289X/C8285X的中断结构中就有 8 个中断级。

中断级序号 0~7 仅仅是一个标号，并不直接表示优先级。相对优先级由中断优先级寄存器 IPR 中的设置决定。IPR中，中断组和子分组结构更细化了各个中断级之间的优先级定义。

#### 5.1.2 中断向量 (VECTORS)

每一个中断级中可以包括一个或多个中断向量，又或者没有任何中断地址。每个中断级最多支持 128 个中断向量（但 S3C8- 系列产品中实际用到的向量个数往往远小于 128）。当一个中断级有多个中断地址时，优先级由硬件决定。S3C828BX/C8289X/C8285X 里有 18 个中断向量。

#### 5.1.3 中断源 (SOURCES)

中断源可以是任何产生中断的外设。可以是外部管脚或者计数器溢出。多个中断源可以共用一个中断向量。S3C828BX/C8289X/C8285X的中断结构中有 18 个潜在的中断源。

当中断服务程序开始之后，必须清除相应的中断标志位，如果不是硬件自动清零，就必须软件手动清零。标志位类型是由中断源的标志位产生/清除机制决定的。

### 5.1.4 中断类型

之前介绍的 S3C8 系列中断结构的三个组成部分 — 中断级，中断向量和中断源 — 充分使用了芯片的中断逻辑，一起决定了芯片的中断结构。中断结构一共有三种可能的组合，分别称为 Type1, 2, 3。三者之间的区别在于分配给每个中断级的中断向量和中断源的个数不同。（详见 [图 5-1](#)）

- Type 1: 一个中断级 (IRQn) + 一个中断向量 ( $V_1$ ) + 一个中断源 ( $S_1$ )
- Type 2: 一个中断级(IRQn) + 一个中断向量 ( $V_1$ ) + 多个中断源( $S_1 - S_n$ )
- Type 3: 一个中断级(IRQn) + 多个中断向量( $V_1 - V_n$ ) + 多个中断源 ( $S_1 - S_n, S_{n+1} - S_{n+m}$ )

S3C828BX/C8289X/C8285X只使用了上述两种 Type。

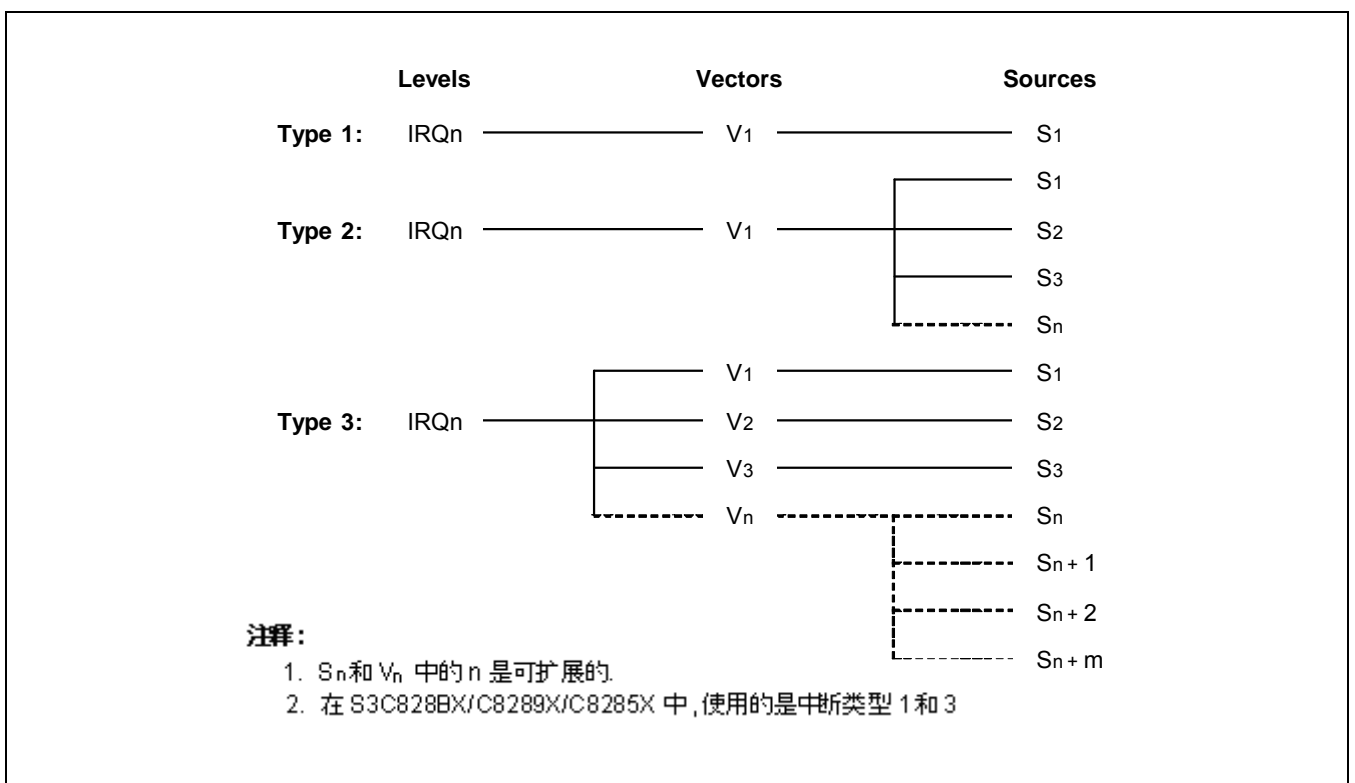


图 5-1 S3C8系列中的中断类型



### 5.1.5 S3C828BX/C8289X/C8285X 中断结构

S3C828BX/C8289X/C8285X支持 18 个中断源，每一个中断源都有对应的中断向量地址。在这款芯片中，CPU 可以响应 8 个中断级（参见图 5-2）。

多个中断级同时发出中断请求时，中断优先级寄存器（IPR）中的值决定了哪一个竞争中的中断级先被 CPU 响应。如果同一个中断级中的多个中断源同时发出中断请求，中断向量地址最小的那个中断源拥有最高优先级，首先被响应（同一中断级内中断源优先级顺序由硬件决定）。

当 CPU 响应某个中断请求以后，中断处理就开始了。此时所有的其他中断都处于禁止状态，PC 和 FLAGS 的值被压入堆栈。从中断向量地址中获取中断服务程序的起始地址（16 位地址由中断向量地址处的 8 位和向量地址之后的 8 位数据一起构成），程序跳转而后开始执行中断服务程序。

Levels	Vectors	Sources	Reset/Clear
RESET	100H	Basic Timer Overflow	H/W
IRQ0	DCH	Timer A match/capture	S/W
	DEH	Timer A overflow	H/W,S/W
IRQ1	E0H	Timer B match	H/W
IRQ2	E2H	Timer 0 match	S/W
IRQ3	E4H	Timer 1 match/capture	S/W
	E6H	Timer 1 overflow	H/W,S/W
IRQ4	E8H	SIO interrupt	S/W
IRQ5	EAH	UART data transmit	S/W
	ECH	UART data receive	S/W
	EEH	Watch timer overflow	S/W
IRQ6	F0H	P0.0 External interrupt	S/W
	F2H	P0.1 External interrupt	S/W
	F4H	P0.2 External interrupt	S/W
	F6H	P0.3 External interrupt	S/W
IRQ7	F8H	P0.4 External interrupt	S/W
	FAH	P0.5 External interrupt	S/W
	FCH	P0.6 External interrupt	S/W
	FEH	P0.7 External interrupt	S/W

**注释：**

1. 在给定的中断级中，越低的中断向量地址中断优先级越高  
例如：在IRQ0中，DCH比DEH优先级高。
2. 外部中断由上升沿或下降沿触发，触发方式由相应的控制寄存器决定。

图 5-2 S3C828BX/C8289X/C8285X 中断结构

### 5.1.6 中断向量地址

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 中断结构的所有中断向量地址都位于内部 64K 字节程序存储空间, 0H–FFFFH, 或 16, 32K 字节的向量地址区(详见图 5-3)。

没有被用作中断向量的地址空间可以当作普通的程序存储空间。此时要小心千万不能与中断向量区交叠 (表 5-1 列出了所有的向量地址)。

ROM 中的程序复位地址为 0100H。

S3F828BX ROM (Full Flash 产品) 中的复位地址可在 smart option 中修改。详情参见第 19 章 嵌入式 Flash 存储器接口。

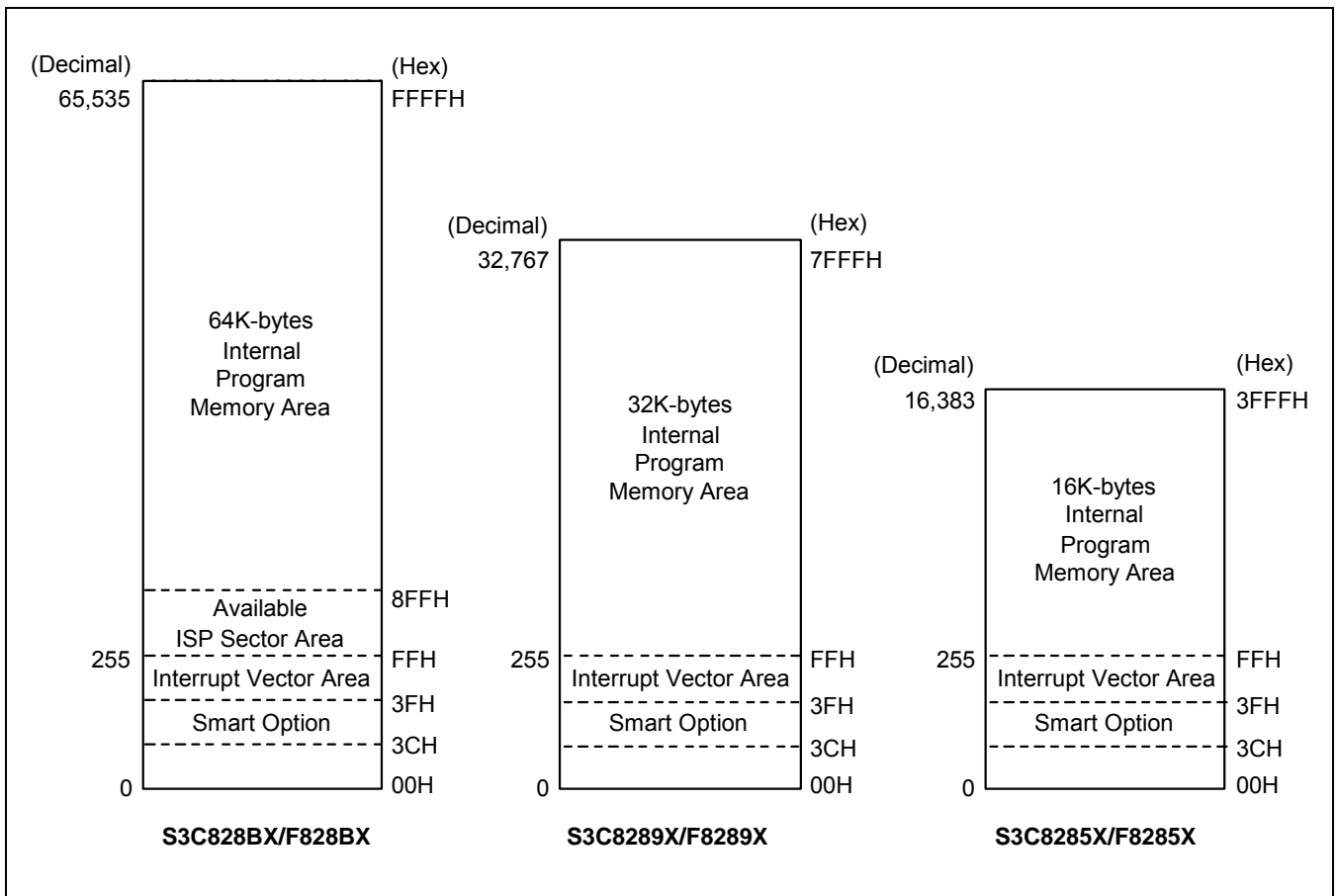


图 5-3 ROM 中断向量地址区

表 5-1 中断向量

向量地址		中断源	中断请求		复位/清除	
十进制值	十六进制值		中断优先级	级内优先级	硬件	软件
256	100H	Basic timer 溢出	Reset	–	√	
220	DCH	Timer A 匹配/捕获	IRQ0	0		√
222	DEH	Timer A 溢出		1	√	√
224	E0H	Timer B 匹配	IRQ1	–	√	
226	E2H	Timer 0 匹配	IRQ2	–		√
228	E4H	Timer 1 匹配/捕获	IRQ3	0		√
230	E6H	Timer 1 溢出		1	√	√
232	E8H	SIO 中断	IRQ4	–		√
234	EAH	UART 数据发送中断	IRQ5	0		√
236	ECH	UART 数据接收中断		1		√
238	EEH	钟表定时器 溢出		2		√
240	F0H	P0.0 外部中断	IRQ6	0		√
242	F2H	P0.1 外部中断		1		√
244	F4H	P0.2 外部中断		2		√
246	F6H	P0.3 外部中断		3		√
248	F8H	P0.4 外部中断	IRQ7	0		√
250	FAH	P0.5 外部中断		1		√
252	FCH	P0.6 外部中断		2		√
254	FEH	P0.7 外部中断		3		√

## 注释:

1. 中断优先级标注是反向的: "0" 的优先级最高, "1" 第二高, 依次往下。
2. 若 2 个或多个同级中断同时发生, 则向量地址较低的中断具有较高的优先级。同级中断的优先级由硬件决定。
3. Timer A 或 Timer 1 的两个中断不能同时响应, 因此只能使用其中的一个中断源。

### 5.1.7 使能/禁止中断的指令 (EI, DI)

执行使能中断指令 (EI) 使能全局中断结构。然后，所有的中断按照既定的优先级顺序执行。

**注释：** 复位后，在初始化程序中应该包含 EI 指令来使能全局中断结构。

在正常的操作中，可以执行 DI（禁止中断）指令随时禁止全局中断处理。EI 和 DI 指令会改变寄存器 SYM 的第 0 位状态。

### 5.1.8 系统级中断控制寄存器

除了特定中断源的控制寄存器外，4 个系统级寄存器也控制中断处理：

- 中断屏蔽寄存器，IMR，使能（解除屏蔽）或者 禁止（屏蔽）中断级。
- 中断优先级控制寄存器，IPR，控制各个中断级之间的相对优先级。
- 中断请求寄存器，IRQ，包含每个中断级的中断标志位（不同于中断源的标志位）。
- 系统模式控制寄存器，SYM，使能或者禁止全局中断处理（SYM 的设置还可以使能快速中断并在硬件支持的情况下控制外部接口）。

表 5-2 中断控制寄存器描述

控制寄存器	ID	R/W	功能描述
中断屏蔽寄存器	IMR	R/W	IMR 寄存器中的位设置可以使能或禁止 IRQ0–IRQ7 中任意一个中断级的中断处理。
中断优先级寄存器	IPR	R/W	控制各个中断级间的相对优先级。S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X中的 7 个中断级被分作 3 个组：A, B 和 C 组。A组包括 IRQ0 和 IRQ1，B组包括IRQ2, IRQ3 和 IRQ4，C 组包括 IRQ5, IRQ6 和 IRQ7。
中断请求寄存器	IRQ	R	包含每个中断级的中断标志位
系统模式控制寄存器	SYM	R/W	使能或者禁止快速中断处理，动态全局中断处理

**注释：** IMR 寄存器改变成任何值前，都必须禁止所有中断。建议使用 DI 指令。

### 5.1.9 中断处理控制要点

中断处理控制可以通过两种方式来完成：全局或者特定中断级和中断源控制。系统级的中断结构控制要点如下：

- 全局中断的使能/禁止 (通过 EI 和 DI 指令或直接操作 SYM.0)
- 中断级使能/禁止 (通过寄存器 IMR)
- 中断级优先级设置 (通过寄存器 IPR)
- 通过相应的外设控制寄存器使能/禁止中断源

**注释：** 包含中断处理的应用程序中，务必确保包含必须的寄存器文件地址 (寄存器指针) 信息。

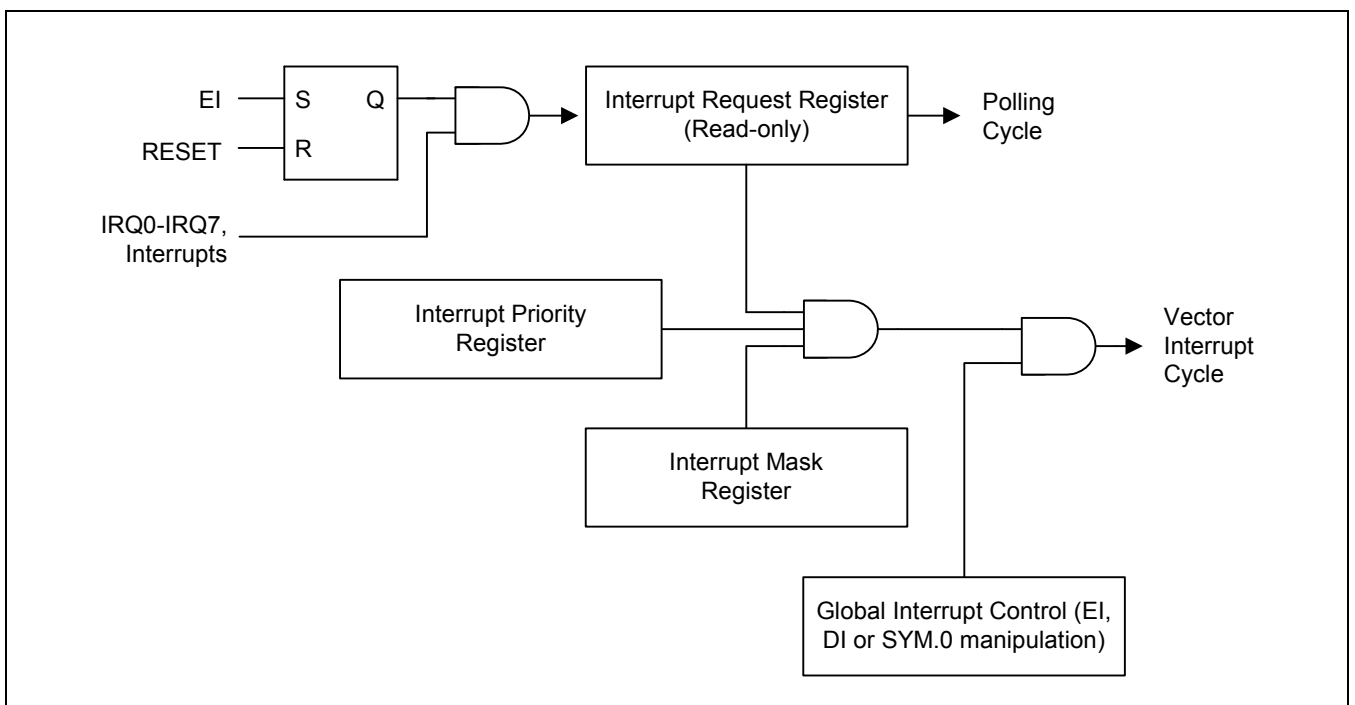


图 5-4 中断功能框图

## 5.1.10 外设中断控制寄存器

每一个中断源都有一个或多个对应的外设控制寄存器来控制相关外设产生的中断。(详见表 5-3)

表 5-3 中断源控制和数据寄存器

中断源	中断级	寄存器	Set 1 寄存器地址
Timer A 匹配/捕获 Timer A 溢出	IRQ0	TACON TACNT TADATA	E8H, bank 0 F9H, bank 0 EAH, bank 0
Timer B 匹配	IRQ1	TBCON TBDATAH, TBDATAL	F2H, bank 0 F0H, F1H, bank 0
Timer 0 匹配	IRQ2	T0CON T0CNTH, T0CNTL, T0DATAH, T0DATAL	E3H, bank 0 E4H, E5H, bank 0 E6H, E7H, bank 0
Timer 1 匹配/捕获 Timer 1 溢出	IRQ3	T1CON T1CNTH, T1CNTL T1DATAH, T1DATAL	EBH, bank 0 ECH, EDH, bank 0 EEH, EFH, bank 0
SIO 中断	IRQ4	SIOCON SIODATA SIOPS	E0H, bank 0 E1H, bank 0 E2H, bank 0
UART 数据发送中断 UART 数据接收中断  钟表定时器溢出	IRQ5	UARTCON UDATA BRDATA WTCON	F6H, bank 0 F7H, bank 0 F8H, bank 0 D1H, bank 0
P0.0 外部中断 P0.1 外部中断 P0.2 外部中断 P0.3 外部中断	IRQ6	P0CONL P0INTL P0PND	E1H, bank 1 E3H, bank 1 E4H, bank 1
P0.4 外部中断 P0.5 外部中断 P0.6 外部中断 P0.7 外部中断	IRQ7	P0CONH P0INTH P0PND	E0H, bank 1 E2H, bank 1 E4H, bank 1

**注释:** 若某一中断在 IMR 寄存器中未被屏蔽 (使能中断级), 则其中断标志位和使能位必须在 DI 指令之后写入。

### 5.1.11 系统模式寄存器 (SYM)

系统模式寄存器 SYM (地址: DEH, Set 1) 可以用来使能/禁止全局中断处理, 控制快速中断处理。参见图 5-5)

复位时清0 SYM.1 和 SYM.0, 快速中断级选择位 SYM.4–SYM.2 的值不确定。

EI 和 DI 指令可以使能或禁止全局中断处理, 相应的, 也可以通过修改 SYM.0 得到同样效果。为使能中断处理, 复位后应该在初始化程序中包含 EI 指令。尽管可以通过操作 SYM.3 位直接使能或禁止中断, 但还是建议用 EI 指令或 DI 指令。

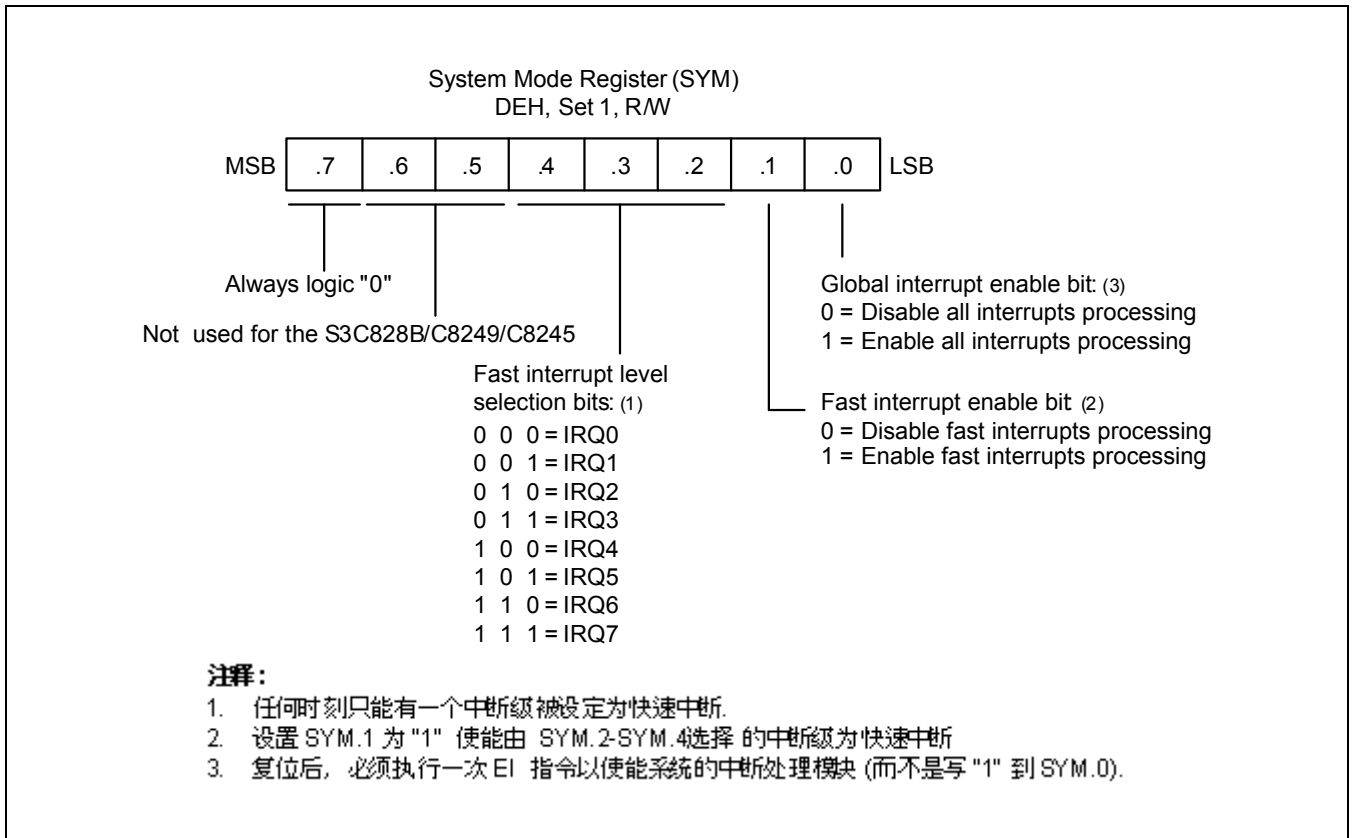


图 5-5 系统模式寄存器 (SYM)



### 5.1.12 中断屏蔽寄存器 (IMR)

中断屏蔽寄存器 IMR (地址: DDH, Set 1) 用来使能或禁止各个中断级的中断处理。复位后, IMR 的所有位都是不确定的, 所以必须在初始化程序中根据应用设置该寄存器。

IMR 中的每一位都对应一个中断级, 第 1 位对应于 IRQ1, 第 2 位对应于 IRQ2, 以此类推。当 IMR 中的某位被清“0”后, 与之对应的中断级的中断处理就被禁止(屏蔽)了。当设置为“1”时, 相应中断级的中断处理就被使能(解除屏蔽)了。

IMR 寄存器映射于 set 1 的 DDH 地址。可通过指令使用寄存器寻址模式对每一位进行读写。

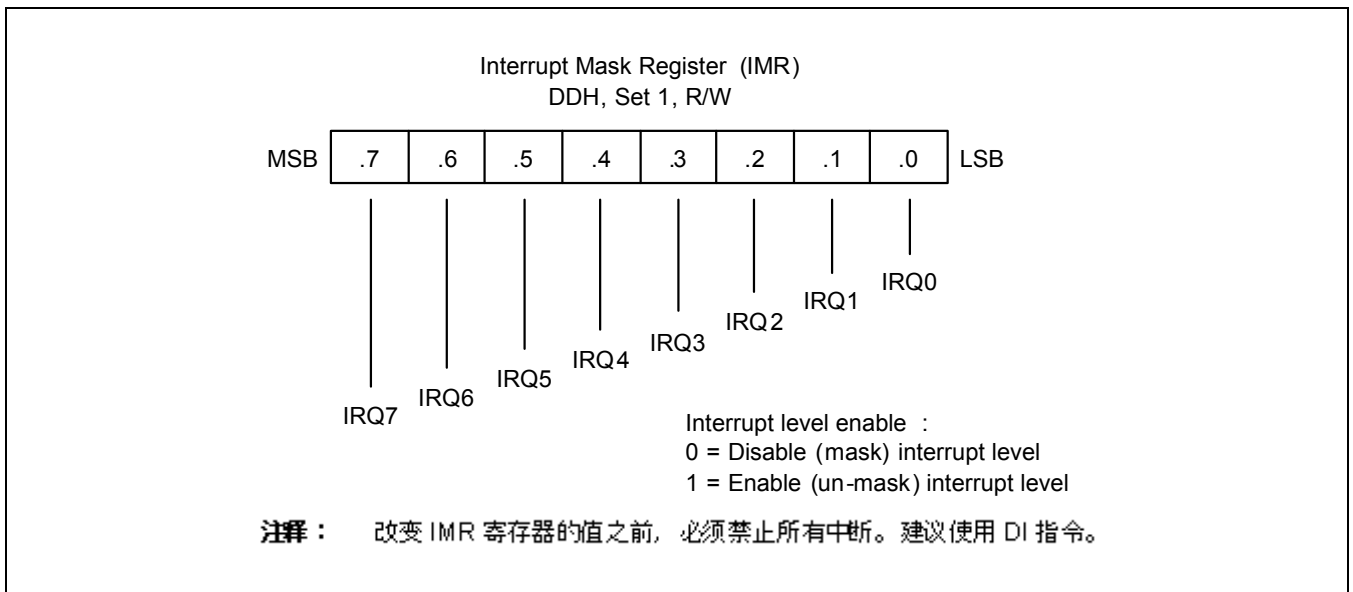


图 5-6 中断屏蔽寄存器(IMR)

### 5.1.13 中断优先级寄存器 (IPR)

中断优先级控制寄存器 IPR (地址: FFH, Set 1, Bank 0) 可以用来设置 MCU 中断结构中各个中断级的相对优先级。复位后, IPR 的所有位都是不确定的, 所以必须在初始化程序中根据应用设置该寄存器。

当多个中断源同时发出中断请求时, 优先级最高的那个中断源首先被 CPU 响应。如果同一中断级中的多个中断源同时发出中断请求, 中断向量地址最小的那个最先被响应。

为支持相对中断优先级编程, 中断逻辑将中断级分为组和子组。请注意, 这些组 (子组) 仅供 IPR 逻辑定义 IP 寄存器的优先级。(详见 [图 5-7](#)):

- Group A     IRQ0, IRQ1
- Group B     IRQ2, IRQ3, IRQ4
- Group C     IRQ5, IRQ6, IRQ7

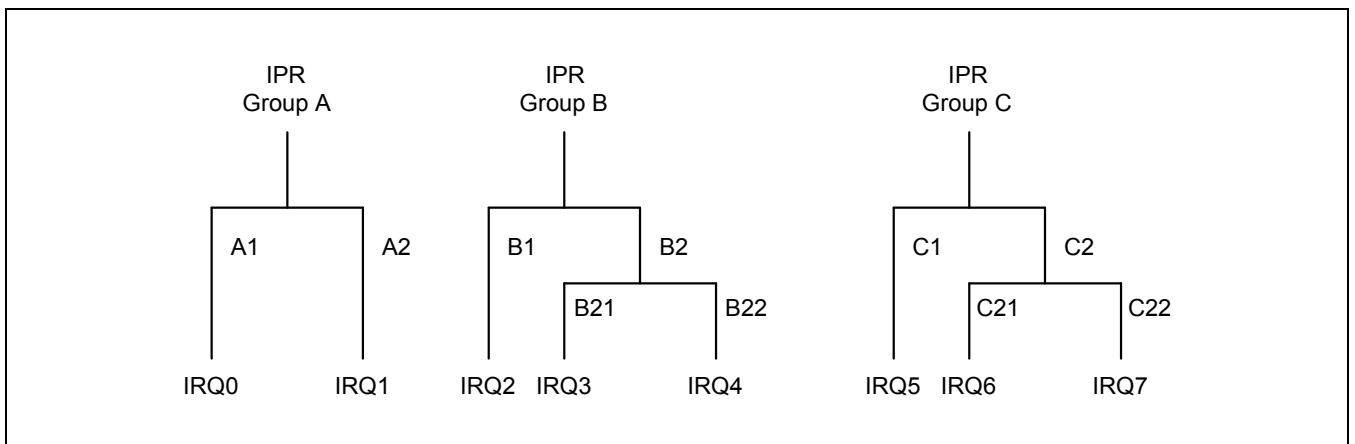


图 5-7 中断请求优先级组

如 [图 5-8](#) 所示, IPR.7, IPR.4 和 IPR.1 控制中断组 ABC 的相对优先级。例如, 当这3位设置为“001B”时, 这3个中断组的优先级顺序为 B > C > A; 设置为“101B”时, 这3个中断组的优先级顺序为 C > B > A。

IPR 其他位控制功能描述如下:

- IPR.5 控制中断组C 内中断级的相对优先级。
- 中断组C 包含一个子组, 内部的中断级5, 6, 7有额外的优先级。子组的优先级由 IPR.6 的值决定。IPR.5 控制中断组C。
- IPR.0 控制 IRQ0 和 IRQ1 的相对优先级。

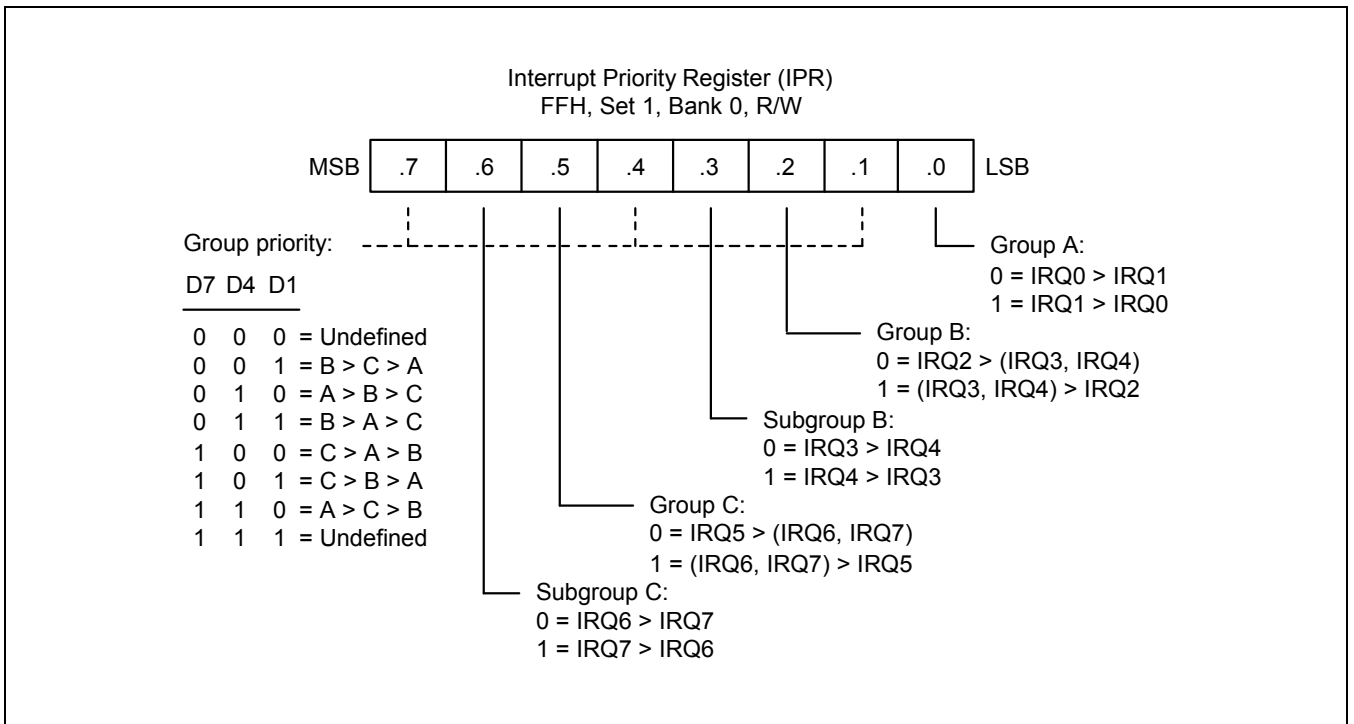


图 5-8 中断优先级寄存器 (IPR)

### 5.1.14 中断请求寄存器 (IRQ)

可以通过查询中断请求寄存器 IRQ (地址: DCH, Set 1) 的每一位监视所有中断级的中断请求情况。寄存器中的每一位都对应于一个同序号的中断级: 第 0 位对应于 IRQ0, 第 1 位对应于 IRQ1, 以此类推。“0”代表目前该中断级没有中断发生, “1”代表该中断级中的某个中断源发出了中断请求。

IRQ 寄存器是只读的。可随时使用位或字节方式访问 IRQ 寄存器来读取 (测试) 某一中断级当前的中断请求。复位后, 所有状态位清0。

即使是执行 DI 指令后, 即全局中断处理禁止的情况下, 仍然可以查询 IRQ 的内容。此时如果有中断发生, CPU 并不会做出任何响应。但是仍然可以通过查询 IRQ 确定中断发生情况。可在全局中断屏蔽的情况下, 使用上述方法判断事件的发生。

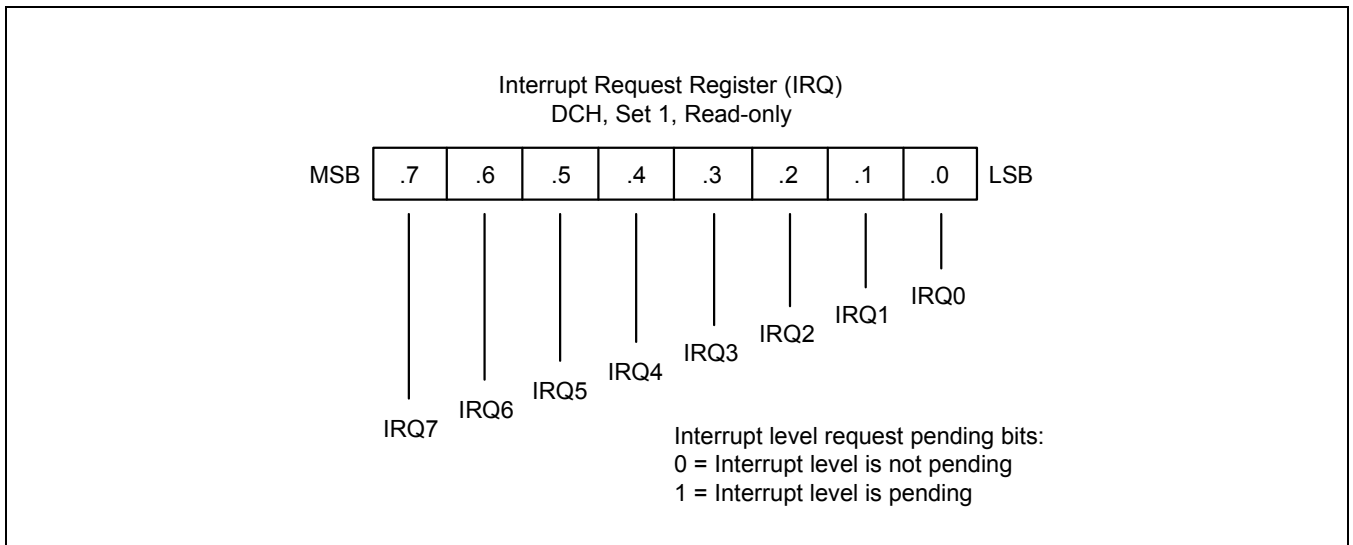


图 5-9 中断请求寄存器 (IRQ)

### 5.1.15 中断标志位类型

#### 5.1.15.1 概述

有两种类型的中断标志位：一种会在中断服务程序执行完毕后硬件自动清零，一种必须在中断服务程序中软件清零。

#### 5.1.15.2 硬件自动清零的标志位

对于硬件自动清零的标志位来说，当中断请求时，中断逻辑置高相应的标志位。然后触发 IRQ 脉冲告知 CPU 有一个中断正等待处理。CPU 随后发送 IACK 给中断源确认接受请求，然后执行服务程序，最后清除标志位。这种类型的标志位没有被映射，所以不能在软件中对其进行读写。

在S3C828BX/C8289X/C8285X的中断结构中，timer A 溢出中断 (IRQ0)，timer B 匹配中断 (IRQ1)，timer 1 溢出中断 (IRQ3)就属于这个类型的中断，标志位硬件自动清零。

#### 5.1.15.3 中断服务程序中清零的标志位

第二类标志位必须用软件清零。中断服务程序必须在中断返回 (IRET) 前清除标志位。可通过对中断源的模式或控制寄存器中的相应标志位清“0”来实现。

#### 编程实例 5-1 如何清除中断标志位

如下例所示，应使用 load 指令来清除中断标志位。

例：

```

1.          SB1
           LD      POPND, #11111011B      ; P0.2中断标志为清0
           •
           •
           •
           IRET

2.          SB0
           LD      INTPND, #11111101B    ; 清除 timer A 匹配/捕获中断标志位
           •
           •
           •
           IRET

```

### 5.1.16 中断源查询顺序

中断请求查询和响应的步骤如下：

1. 某一中断源通过置位中断请求位产生中断请求。
2. CPU 查询程序识别该中断源的中断挂起状态。
3. CPU 检查该中断源的中断级。
4. CPU 产生中断确认信号。
5. 中断逻辑决定中断的向量地址。
6. 中断服务程序启动并将该中断源的标志位清0。(硬件清0或软件清0)
7. CPU 继续查询中断请求。

### 5.1.17 中断服务程序

中断处理前，必须满足以下条件：

- 全局中断处理使能 (EI)
- 该中断源所处的中断级处于使能状态(IMR 寄存器)
- 若有多个中断请求，该中断级必须具有最高优先级
- 该中断源处于使能状态 (外设控制寄存器)

当所有的条件都满足之后，中断请求会在指令周期的最后被确认。随后，CPU 将开始中断处理并完成下列操作：

1. SYM 寄存器的 SYM.0 位清零，以禁止所有后续的中断。
2. 把程序计数器 PC 和状态寄存器压入堆栈。
3. 跳转到中断向量，获取中断服务程序地址。
4. 跳转执行中断服务程序

中断服务程序完成以后，CPU 发出中断返回指令 (IRET)。中断返回指令 IRET 会将堆栈中的 PC 和状态寄存器重置，同时置位SYM.0，使 CPU 能响应后续的中断请求。

### 5.1.18 中断向量地址的生成

ROM (00H~FFH) 中的中断向量区域，包含了中断结构中的每一级对应的中断服务程序的入口地址。向量化的中断处理流程如下：

1. 程序计数器 PC 低字节压入堆栈
2. 程序计数器 PC 高字节压入堆栈
3. FLAGS 寄存器压入堆栈
4. 从中断向量地址中取出中断服务程序入口地址 (高字节)
5. 从中断向量地址中取出中断服务程序入口地址 (低字节)
6. 跳转执行 16 位中断向量地址所确定的中断服务程序

**注释：** 16 位向量地址始终起始于 ROM 空间 00H ~ FFH 中的某个偶数地址。

### 5.1.19 中断嵌套

可以在一个低优先级中断请求的处理过程中，嵌套一个高优先级的中断请求。为此，必须遵循如下步骤：

1. 将当前的8位 IMR 寄存器压栈。
2. 重置 IMR，仅使能那个有较高优先级的中断。
3. 执行 EI 使能中断处理 (较高优先级的中断一旦发生就可被响应)
4. 在较低优先级的中断服务处理程序最后，将 IMR 寄存器弹栈，恢复进入中断前的状态。
5. 执行中断返回指令 IRET。

可根据不同的应用，简化上述步骤。

### 5.1.20 指令指针 (IP)

S3C8- 全系列都采用了指令指针 (IP) 以支持高速中断处理，又称作快速中断 (fast interrupts) 。IP实际是个寄存器对，高 8 位 IPH (IP15~IP8) 和低 8 位 IPL (IP7~IP0) 分别位于地址单元 DAH 和 DBH。

### 5.1.21 快速中断处理

一般的中断处理需要 16 个 CPU 时钟周期，而一旦将给定中断级设为快速中断，只需要大概 6 个 CPU 时钟周期就可以完成整个中断处理。通过写 SYM.4–SYM.2 可以将快速中断处理应用于某一个中断级。然后需要置位 SYM.1 使能快速中断处理。

快速中断处理用到其他2个系统寄存器：

- 指令指针 (IP) 包含了快速中断服务程序的起始地址（中断处理后会和 PC 交换完成中断返回）
- 当快速中断发生时，FLAGS 寄存器的内容会自动载入一个没有地址映射的专用寄存器，称为 FLAGS'

**注释：** 对于 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X，8 个中断级 IRQ0–IRQ7 中的任何一个都可以选择为快速中断处理的对象。

#### 5.1.21.1 快速中断初始化步骤

以下步骤初始化快速中断：

1. 将中断服务程序的起始地址载入指令指针 (IP)。
2. 将中断级编号 (IRQn) 载入快速中断选择区 (SYM.4–SYM.2)。
3. 在 SYM 寄存器的快速中断使能位中写 "1"。

#### 5.1.21.2 快速中断服务程序

当指定为快速中断的中断级中出现中断请求时，如下事件发生：

1. PC 和 IP 交换内容
2. FLAGS 寄存器值载入 FLAGS'
3. FLAGS 寄存器中的快速中断状态位置高
4. 开始中断服务
5. 若快速中断标志位被置高，当中断服务程序结束后，PC 和 IP 再次交换内容
6. FLAGS' 寄存器的值自动复制回 FLAGS 寄存器
7. FLAGS 寄存器中的快速中断标志位自动清0



### 5.1.21.3 中断标志位类型间关系

如前所述，有2种中断标志位：1种在中断服务程序被识别并执行之后由硬件自动清0；另一种必须由中断服务程序软件清0。可为快速中断处理选择任一种类型的中断标志位清0 – 硬件清0或软件清0。

### 5.1.21.4 编程指导

切记使能/禁止快速中断的唯一方法是置位/清0 SYM 寄存器中的快速中断使能位 SYM.1。执行 EI 或 DI 指令使能或禁止全局中断处理，包括快速中断。若使用快速中断，务必要在快速中断服务程序的最后，在IP中写入一个新的起始地址。

# 6 指令集

## 6.1 概述

SAM8RC 指令集支持对大容量寄存器卷的操作，一共包含 78 条指令，具有强大的数据处理能力。指令集特性如下：

- 实现了所有 8 位算术和逻辑操作，包括乘法和除法
- 没有专门的输入/输出指令（输入/输出控制寄存器和数据寄存器直接映射到寄存器卷中）
- 十进制调整，包括 BCD 操作
- 16 位(字)数据可自增或自减
- 灵活的位寻址、循环和移位指令

### 6.1.1 数据类型

SAM8 CPU 可以实现位操作、字节操作、BCD 数字操作和双字节操作。寄存器的每个位可以被置 1、清 0、取反和测试。一个字节的各位按从 7 到 0 编号，其中 0 位是最低位 (在最右边)。

### 6.1.2 寄存器访问

为访问寄存器，应指定寄存器卷中地址为 0 – 255 之间的 8 位地址或工作寄存器的 4 位地址。工作寄存器中，寄存器对可以访问 16 位程序存储空间和数据存储空间。关于寄存器访问的详细描述，请参考第 2 章“地址空间”。

### 6.1.3 寻址模式

有 7 种寻址模式：寄存器寻址(R)，间接寄存器寻址(IR)，偏址寻址(X)，直接寻址(DA)，相对地址寻址(RA)，立即数寻址(IM)和间接寻址(IA)。有关寻址模式的详细描述，请参考第 3 章“寻址模式”。

表 6-1 指令集简介

助记符	操作数	指令介绍
<b>数据传送类指令</b>		
CLR	dst	清零
LD	dst,src	传送数据
LDB	dst,src	传送位数据
LDE	dst,src	传送数据（访问外部数据存储空间）
LDC	dst,src	传送数据（访问程序存储空间）
LDED	dst,src	传送数据后地址减 1（访问外部数据存储空间）
LDCD	dst,src	传送数据后地址减 1（访问程序存储空间）
LDEI	dst,src	传送数据后地址加 1（访问外部数据存储空间）
LDCI	dst,src	传送数据后地址加 1（访问程序存储空间）
LDEPD	dst,src	传送数据前地址减 1（访问外部数据存储空间）
LDCPD	dst,src	传送数据前地址减 1（访问程序存储空间）
LDEPI	dst,src	传送数据前地址加 1（访问外部数据存储空间）
LDCPI	dst,src	传送数据前地址加 1（访问程序存储空间）
LDW	dst,src	传送字数据
POP	dst	出栈
POPUD	dst,src	弹出用户栈（减 1）
POPUI	dst,src	弹出用户栈（加 1）
PUSH	src	压栈
PUSHUD	dst,src	压入用户栈（减 1）
PUSHUI	dst,src	压入用户栈（加 1）
<b>算术操作类指令</b>		
ADC	dst,src	带进位加法
ADD	dst,src	不带进位加法
CP	dst,src	比较指令
DA	dst	十进制调整
DEC	dst	字节减 1
DECW	dst	字减 1
DIV	dst,src	除法指令
INC	dst	字节加 1
INCW	dst	字加 1
MULT	dst,src	乘法指令
SBC	dst,src	带借位减法

助记符	操作数	指令介绍
SUB	dst,src	不带借位减法
<b>逻辑操作类指令</b>		
AND	dst,src	逻辑与
COM	dst	取反
OR	dst,src	逻辑或
XOR	dst,src	逻辑异或
<b>程序控制指令</b>		
BTJRF	dst,src	位测试, 如果为 0 跳转
BTJRT	dst,src	位测试, 如果为 1 跳转
CALL	dst	调用子程序
CPIJE	dst,src	比较, 如果相等跳转
CPIJNE	dst,src	比较, 如果不等跳转
DJNZ	r,dst	寄存器减 1, 不为 0 跳转
ENTER		进入
EXIT		跳出
IRET		中断返回
JP	cc,dst	有条件跳转
JP	dst	无条件跳转
JR	cc,dst	有条件相对跳转
NEXT		<b>Next</b> 指令
RET		子程序返回
WFI		等待中断
<b>位操作指令</b>		
BAND	dst,src	位与
BCP	dst,src	位比较
BITC	dst	位取反
BITR	dst	位清零
BITS	dst	位置 1
BOR	dst,src	位或
BXOR	dst,src	位异或
TCM	dst,src	取反后位测试
TM	dst,src	位测试指令

助记符	操作数	指令介绍
<b>循环和移位指令</b>		
RL	dst	循环左移
RLC	dst	带进位循环左移
RR	dst	循环右移
RRC	dst	带进位循环右移
SRA	dst	算术右移
SWAP	dst	交换
<b>CPU控制指令</b>		
CCF		进位标志取反
DI		屏蔽全局中断
EI		使能全局中断
IDLE		进入 IDLE 模式
NOP		空操作
RCF		进位标志清零
SB0		选择寄存器块 bank 0
SB1		选择寄存器块 bank 1
SCF		进位标志置 1
SRP	src	设置寄存器指针
SRP0	src	设置寄存器指针 0
SRP1	src	设置寄存器指针 1
STOP		进入 STOP 模式

注释: LDE, LDED, LDEI, LDEPP 和 LDEPI 指令可用来读/写 64 K 字节的外部数据存储空间。

### 6.1.4 标志寄存器(FLAGS)

8 位标志寄存器 FLAGS 描述当前 CPU 的操作状态。其中的 4 位 (FLAGS.4–FLAGS.7) 可以用于测试和条件转移指令；另外两个标志位 FLAGS.3 和标志位 FLAGS.2 用于 BCD 算术操作。

还有一个标志位 FLAGS.1 用于指示快速中断处理；FLAGS.0 表示当前正在寻址的 Bank 地址状态，是 Bank 0 还是 Bank 1。

只要结果不影响到状态位，FLAGS 寄存器可以通过指令（如 LOAD 指令）置 1 或者清 0。逻辑和算术类操作指令，例如与、或、异或、加法和减法操作，会影响到标志位寄存器。例如，AND 指令会根据结果改变 Zero, Sign 和 Overflow (Z、S、O) 标志。如果 AND 指令使用 FLAGS 作为目标寄存器，将会同时发生两次 FLAGS 寄存器的写操作，造成不可预知的后果。

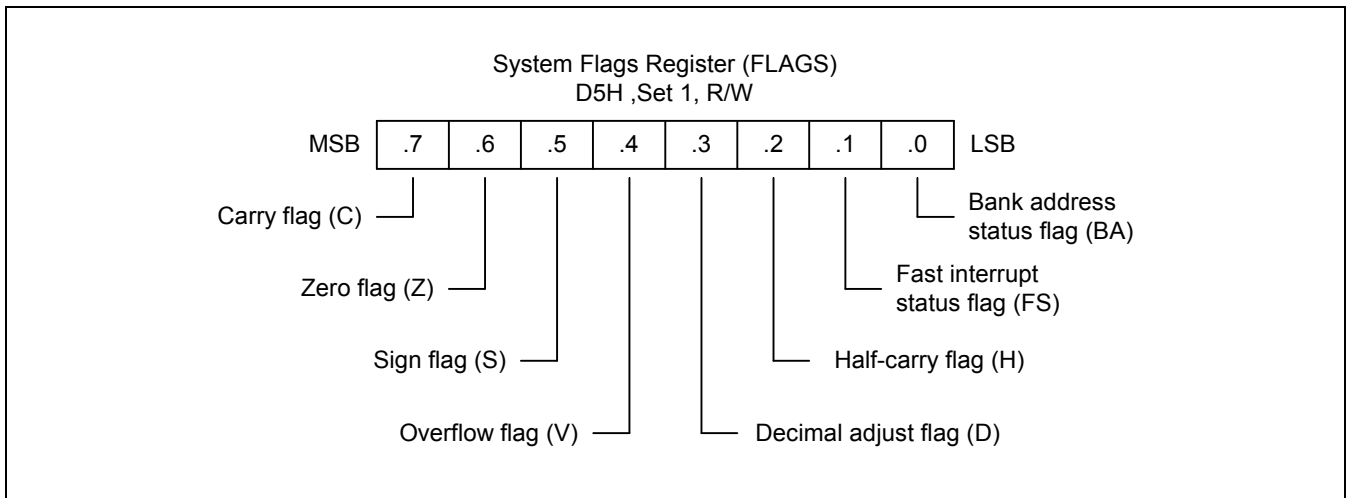


图 6-1 系统标志寄存器 (FLAGS)

## 6.1.5 标志位描述

### 6.1.5.1 C: 进（借）位标志 (FLAGS.7)

如果算术操作后，最高位产生进位或借位，则此标志位为 1。  
移位、循环移位操作之后，此位保存最后移出的那一位值。指令可以对此位置 1、清 0、取反操作。

### 6.1.5.2 Z: 零标志位(FLAGS.6)

如果算术、逻辑操作的结果为 0，则此标志位为 1。  
位测试指令、移位指令、循环移位指令都会影响此标志位，如果结果为逻辑 0，则此标志位为 1。

### 6.1.5.3 S: 符号标志位 (FLAGS.5)

算术、逻辑、循环、移位操作之后，操作结果最高位的状态反映在符号位。逻辑 0 表示操作结果是正数，逻辑 1 表示操作数结果是负数。

### 6.1.5.4 V: 溢出标志 (FLAGS.4)

当操作结果大于 +127 或 小于 -128 时，溢出标志将会置 1。逻辑操作之后，它将会被清 0。

### 6.1.5.5 D: 十进制调整标志 (FLAGS.3)

DA 标志用于指明 BCD 操作中何种类型的指令是最后被执行的，因此随后的十进制调整指令可以正确执行。  
编程者通常不能访问 DA 位，也不能用来做测试的条件码。

### 6.1.5.6 H: 半字节进（借）位标志 (FLAGS.2)

当加法操作时第 3 位产生进位或减法操作时向第 4 位发生借位，H 标志将被置 1。  
通常用在 DA 指令中，将前一次的加法或减法结果（二进制码）转化为十进制结果（BCD）。程序通常不会直接用到 H 标志。

### 6.1.5.7 FIS: 快速中断状态标志 (FLAGS.1)

在快速中断执行期间，FIS 位被置 1；快速中断返回时，FIS 位被清 0。当 FIS 被置 1 时，将禁止所有中断，直到 IRET 指令被执行后，才重新打开快速中断。

### 6.1.5.8 BA: 寄存器块地址标志 (FLAGS.0)

BA 标志表明内部寄存器卷 Set 1 中哪个寄存器块被选中，是 bank 0 还是 bank 1。当执行 SB0 指令，BA 标志被清零（选择 bank 0）；当执行 SB1 指令，BA 标志被置 1（选择 bank 1）。

## 6.1.6 指令集符号

表 6-2 标志位符号

标志位	描述
C	进（借）位标志
Z	零标志
S	符号标志
V	溢出标志
D	十进制调整标志
H	半字节进（借）位标志
0	清为逻辑 0
1	置为逻辑 1
*	根据相应操作置 1 或清 0
-	不受影响
x	不确定

表 6-3 指令集符号

符号	描述
dst	目的操作数
src	源操作数
@	间接寄存器地址前缀
PC	程序计数器
IP	指令指针
FLAGS	标志寄存器(D5H)
RP	寄存器指针
#	立即数或寄存器访问的前缀
H	十六进制后缀
D	十进制后缀
B	二进制后追
opc	指令代码



表 6-4 指令符号的定义

符号	描述	实际操作范围
cc	条件码	参考表格 6-6
r	工作寄存器	Rn (n = 0–15)
rb	工作寄存器的位 b	Rn.b (n = 0–15, b = 0–7)
r0	工作寄存器的位 0, 最低位	Rn (n = 0–15)
rr	工作寄存器对	RRp (p = 0, 2, 4, ..., 14)
R	寄存器或者工作寄存器	reg or Rn (reg = 0–255, n = 0–15)
Rb	寄存器或者工作寄存器的位 b	reg.b (reg = 0–255, b = 0–7)
RR	寄存器对或工作寄存器对	reg or RRp (reg = 0–254, 只能是偶数 p = 0, 2, ..., 14)
IA	间接寻址模式	addr (addr = 0–254, 只能是偶数)
lr	间接工作寄存器寻址	@Rn (n = 0–15)
IR	间接工作寄存器寻址或间接寄存器寻址	@Rn or @reg (reg = 0–255, n = 0–15)
lrr	间接工作寄存器对	@RRp (p = 0, 2, ..., 14)
IRR	间接寄存器对或间接工作寄存器对	@RRp or @reg (reg = 0–254, 只能是偶数 p = 0, 2, ..., 14)
X	基址寻址模式	#reg [Rn] (reg = 0–255, n = 0–15)
XS	短偏址寻址模式	#addr [RRp] (addr = 范围 –128 to +127, p = 0, 2, ..., 14)
xl	长偏址寻址模式	#addr [RRp] (addr = 范围 0–65535, p = 0, 2, ..., 14)
da	直接寻址模式	addr (addr = 范围 0–65535)
ra	相对地址寻址模式	addr (addr = 在 +127 到 -128 范围内的数字)
im	立即数寻址模式	#data (data = 0–255)
iml	长立即数寻址模式	#data (data = 范围 0–65535)

表 6-5 指令代码快速参考

指令代码对照图									
LOWER NIBBLE (HEX)									
	-	0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,lr2	ADD R2,R1	ADD IR2,R1	ADD R1,IM	BOR r0-Rb
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,lr2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	BCP r1.b, R2
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,lr2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	BXOR r0-Rb
E	3	JP IRR1	SRP/0/1 IM	SBC r1,r2	SBC r1,lr2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	BTJR r2.b, RA
R	4	DA R1	DA IR1	OR r1,r2	OR r1,lr2	OR R2,R1	OR IR2,R1	OR R1,IM	LDB r0-Rb
	5	POP R1	POP IR1	AND r1,r2	AND r1,lr2	AND R2,R1	AND IR2,R1	AND R1,IM	BITC r1.b
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,lr2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	BAND r0-Rb
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,lr2	TM R2,R1	TM IR2,R1	TM R1,IM	BIT r1.b
B	8	DECW RR1	DECW IR1	PUSHUD IR1,R2	PUSH UI IR1,R2	MULT R2,RR1	MULT IR2,RR1	MULT IM,RR1	LD r1, x, r2
B	9	RL R1	RL IR1	POPUD IR2,R1	POPUI IR2,R1	DIV R2,RR1	DIV IR2,RR1	DIV IM,RR1	LD r2, x, r1
L	A	INCW RR1	INCW IR1	CP r1,r2	CP r1,lr2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, lrr2, xL
E	B	CLR R1	CLR IR1	XOR r1,r2	XOR r1,lr2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, lrr2, xL
	C	RRC R1	RRC IR1	CPIJE lr,r2,RA	LDC r1,lrr2	LDW RR2,RR1	LDW IR2,RR1	LDW RR1,IML	LD r1, lr2
H	D	SRA R1	SRA IR1	CPIJNE lrr,r2,RA	LDC r2,lrr1	CALL IA1		LD IR1,IM	LD lr1, r2
E	E	RR R1	RR IR1	LDCD r1,lrr2	LDCI r1,lrr2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC r1, lrr2, xs
X	F	SWAP R1	SWAP IR1	LDCPD r2,lrr1	LDCPI r2,lrr1	CALL IRR1	LD IR2,R1	CALL DA1	LDC r2, lrr1, xs

表 6-6 指令代码快速参考 (2)

指令代码对照图									
LOWER NIBBLE (HEX)									
	-	8	9	A	B	C	D	E	F
U	0	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NEXT
P	1	↓	↓	↓	↓	↓	↓	↓	ENTER
P	2								EXIT
E	3								WFI
R	4								SB0
	5								SB1
N	6								IDLE
I	7	↓	↓	↓	↓	↓	↓	↓	STOP
B	8								DI
B	9								EI
L	A								RET
E	B								IRET
	C								RCF
H	D	↓	↓	↓	↓	↓	↓	↓	SCF
E	E								CCF
X	F	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NOP

### 6.1.7 条件码

条件转移指令通常包含 4 位条件转移操作判断(cc)。

这些条件转移操作判断的结果将决定程序的跳转方向。例如，比较操作后的“相等”条件转移，只有在两个操作数相等时该条件转移操作才会跳转。条件码列举在表 6-7。

C、Z、S、V 等标志位用作条件转移判断位。指令将会根据这些标志位决定跳转方向。

表 6-7 条件码

二进制	助记符	描述	标志位设置
0000	F	逻辑假	–
1000	T	逻辑真	–
0111 (1)	C	有进位或借位	C = 1
1111 (1)	NC	无进位或借位	C = 0
0110 (1)	Z	结果为 0	Z = 1
1110 (1)	NZ	结果不为 0	Z = 0
1101	PL	正数	S = 0
0101	MI	负数	S = 1
0100	OV	溢出	V = 1
1100	NOV	没有溢出	V = 0
0110 (1)	EQ	相等	Z = 1
1110 (1)	NE	不相等	Z = 0
1001	GE	大于等于	(S XOR V) = 0
0001	LT	小于	(S XOR V) = 1
1010	GT	大于	(Z OR (S XOR V)) = 0
0010	LE	小于等于	(Z OR (S XOR V)) = 1
1111 (1)	UGE	无符号大于等于	C = 0
0111 (1)	ULT	无符号小于	C = 1
1011	UGT	无符号大于	(C = 0 AND Z = 0) = 1
0011	ULE	无符号小于等于	(C OR Z) = 1

#### 注释:

1. 一次算术操作的结果可能同时影响两个标志位。例如，Z 符号位被置起时，Z、EQ 都为真。但是 ADD 指令操作之后，可能会用到 Z；而 CP 指令操作之后，EQ 可能被用到。
2. 如果操作数涉及到无符号数，必须使用 UGE, ULT, UGT, ULE 等条件代码。

### 6.1.8 指令集描述

本章详细介绍了 S3F8- 系列单片机的指令操作，同时给出了具体的编程实例。为便于参阅和快速查找，在介绍指令时采用了统一的格式。对每条指令，采用了如下的描述方法：

- 指令名称（标号）
- 指令全称
- 源操作数/目的操作数的格式
- 具体指令的解释
- 每条指令操作的具体描述
- 每条指令对标志寄存器的影响
- 指令格式、执行周期和访问模式的详细介绍
- 每条指令的编程实例

## 6.1.8.1 ADC — 带进位加法 (Add with Carry)

ADC dst, src

操作:  $dst \leftarrow dst + src + c$ 

目的操作数加上源操作数和 C 位，所得结果保存在目的操作数。源操作数不受影响。在多字节加法中，该指令允许把低字节的进位加到高字节的加法运算中。

标志位:

- C:** 如果加法运算中产生进位，则此位置 1；否则，清 0
- Z:** 如果运算结果为 0，则该位置 1；否则，清 0
- S:** 如果运算结果为负，则该位置 1；否则，清 0
- V:** 如果运算结果产生溢出，该位置 1；否则，清 0
- D:** 总是被清 0
- H:** 如果运算结果的低4位有进位，该位置 1；否则，清 0

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式	
					dst	src
opc	dst   src	2	4	12	r	r
			6	13	r	lr
opc	src	3	6	14	R	R
			6	15	R	IR
opc	dst	3	6	16	R	IM

编程实例:

假设: R1 = 10H, R2 = 03H, C flag = "1", 寄存器 01H = 20H, 寄存器 02H = 03H, 和寄存器03H = 0AH:

```

ADC R1,R2      → R1 = 14H, R2 = 03H
ADC R1,@R2     → R1 = 1BH, R2 = 03H
ADC 01H,02H    → 寄存器 01H = 24H, 寄存器 02H = 03H
ADC 01H,@02H   → 寄存器 01H = 2BH, 寄存器 02H = 03H
ADC 01H,#11H   → 寄存器 01H = 32H

```

在第一个例子中，目的寄存器R1内容为 10H，进位标志位为 1，源寄存器 R2 为 03H。语句“ADC R1,R2”把03H和进位位(“1”)累加到目的数 10H，结果为 14H，保存在 R1。

## 6.1.8.2 ADD — 加法 (Add)

**ADD**            dst, src

**操作:**            dst ← dst + src

源操作数和目的操作数相加，结果保存在目的操作数，源操作数不受影响。

**标志位:**

- C:**        如果结果的最高位有进位，被置 1；否则，被清 0
- Z:**        如果结果为 0，被置 1；否则，被清 0
- S:**        如果结果是负数，被置 1；否则，被清 0
- V:**        如果有溢出，被置 1，也就是说，两个操作数符号相同，运算结果是相反的符号；  
             否则，被清 0
- D:**        总是被清 0
- H:**        如果结果的低4位有进位，被置 1；否则，被清 0

**格式:**

		字节数	时钟周期	指令代码 (Hex)	寻址模式	
					dst	src
opc	dst   src	2	4	02	r	r
			6	03	r	lr
opc	src	dst		04	R	R
			6	05	R	IR
opc	dst	src		06	R	IM

**编程实例:**

假设:            R1 = 12H, R2 = 03H, 寄存器 01H = 21H, 寄存器 02H = 03H, 寄存器 03H = 0AH:

```

ADD   R1,R2      →   R1 = 15H, R2 = 03H
ADD   R1,@R2     →   R1 = 1CH, R2 = 03H
ADD   01H,02H    →   寄存器 01H = 24H, 寄存器 02H = 03H
ADD   01H,@02H   →   寄存器 01H = 2BH, 寄存器 02H = 03H
ADD   01H,#25H   →   寄存器 01H = 46H

```

在第一个例子中，目的工作寄存器 R1 内容为12H，源工作寄存器 R2 内容为 03H。语句“ADD R1, R2”执行 03H+12H，结果为 15H，保存在寄存器 R1。

## 6.1.8.3 AND — 逻辑与 (Logical AND)

**AND**            dst, src

**操作:**            dst ← dst AND src

源操作数与目的操作数执行逻辑与操作，结果保存在目的操作数。  
只有当两个操作数的对应位都为1时，结果的相应位才是 1；否则，该位为 0。  
源操作数不受影响。

**标志位:**

- C:**        不受影响
- Z:**        如果结果为 0，被置 1；否则，被清 0
- S:**        如果结果是负数，被置 1；否则，被清 0
- V:**        总是被清零
- D:**        不受影响
- H:**        不受影响

**格式:**

		字节数	时钟周期	指令代码 (Hex)	寻址模式	
					dst	src
opc	dst   src	2	4	52	r	r
			6	53	r	lr
opc	src	dst				
		3	6	54	R	R
			6	55	R	IR
opc	dst	src				
		3	6	56	R	IM

**编程实例:**

假设:            R1 = 12H, R2 = 03H, 寄存器 01H = 21H, 寄存器 02H = 03H, 寄存器 03H = 0AH:

```

AND  R1,R2      →   R1 = 02H, R2 = 03H
AND  R1,@R2     →   R1 = 02H, R2 = 03H
AND  01H,02H    →   寄存器 01H = 01H, 寄存器 02H = 03H
AND  01H,@02H   →   寄存器 01H = 00H, 寄存器 02H = 03H
AND  01H,#25H   →   寄存器 01H = 21H

```

在第一个例子中，目的操作数 R1 为 12H，源操作数 R2 为 03H，指令“AND R1, R2”对 03H 和 12H 进行逻辑与操作，结果为 02H，保存在寄存器 R1。



## 6.1.8.4 BAND — 位与 (Bit AND)

**BAND** dst, src.b

**BAND** dst.b, src

**操作:** dst(0) ← dst(0) AND src(b)  
or  
dst(b) ← dst(b) AND src(0)

源操作数（或者目的操作数）的特定位置与目的操作数（或者源操作数）的最低位进行逻辑与操作，结果保存在目的操作数的特定位置。目的操作数的其他位不受影响。源操作数不受影响。

**标志位:**

- C:** 不受影响
- Z:** 如果结果为 0，被置 1；否则，被清 0
- S:** 总是被清零
- V:** 不确定
- D:** 不受影响
- H:** 不受影响

**格式:**

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	dst   b   0	src	3	6	67	r0	Rb
opc	src   b   1	dst	3	6	67	Rb	r0

**注释:** 在 3 字节指令各式的第二个字节中，目的或者源操作数地址是 4 位，位地址是 3 位。

**编程实例:**

假设: R1 = 07H 和寄存器 01H = 05H:

```
BAND R1,01H.1    →    R1 = 06H, 寄存器 01H = 05H
BAND 01H.1,R1    →    寄存器 01H = 05H, R1 = 07H
```

在第一个例子中，源寄存器 01H 的内容是 05H，目的寄存器 R1 的内容是 07H，指令“BAND R1, 01H.1”将源寄存器的位 1 和目的寄存器 R1 的位 0 进行逻辑与操作，结果为 06H，保存在寄存器 R1。

## 6.1.8.5 BCP — 位比较 (Bit Compare)

**BCP**            dst, src.b

**操作:**            dst(0) – src(b)

源操作数的特定位与目的操作数的最低位做比较，如果相等，零标志位被置 1，否则被清 0。两个操作数都不受影响。

**标志位:**

- C:**        不受影响
- Z:**        如果结果为 0，被置 1；否则，被清 0
- S:**        总是被清零
- V:**        不确定
- D:**        不受影响
- H:**        不受影响

**格式:**

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	dst   b   0	src	3	6	17	r0	Rb

**注释:**    在 3 字节指令各式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

**编程实例:**

假设:        R1 = 07H 和寄存器 01H = 01H:

BCP    R1,01H.1    →    R1 = 07H, 寄存器 01H = 01H

如果目的寄存器 R1 内容为 07H，源寄存器 01H 的内容为 01H，指令“BCP R1, 01H.1”比较源寄存器 (01H) 的位 1 和目的寄存器 R1 的位 0。因为两个位是不相等的，所以标志位寄存器的 Z 标志位被清零。

### 6.1.8.6 BITC — 位反 (Bit Complement)

**BITC**            dst.b

**操作:**             $\text{dst}(b) \leftarrow \text{NOT } \text{dst}(b)$

这个指令对目的操作数的特定位取反而不影响其他位。

**标志位:**

- C:**     不受影响
- Z:**     如果结果为 0, 被置 1; 否则, 被清 0
- S:**     总是被清零
- V:**     不确定
- D:**     不受影响
- H:**     不受影响

**格式:**

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst   b   0	2	4	57	rb

**注释:**     在指令的第二个字节中, 目的操作数地址是 4 位, 位地址是 3 位。

**编程实例:**

假设:            R1 = 07H

                  BITC R1.1 → R1 = 05H

工作寄存器 R1 内容为 07H, 指令“BITC R1.1”对 R1 的位 1 取反, 并将结果(05H)保存在 R1。因为结果不是“0”, 所以标志位寄存器的 Z 标志被清零。

**6.1.8.7 BITR — 位清零 (Bit Reset)****BITR**            dst.b**操作:**            dst(b) ← 0

BITR 指令将目的操作数的特定位清零，而不影响其它的位。

**标志位:**        没有任何标志位受影响**格式:**

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst   b   0	2	4	77	rb

**注释:** 在指令的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。**编程实例:**

假设:            R1 = 07H:

BITR R1.1 → R1 = 05H

工作寄存器 R1 的内容是 07H，指令“BITR R1.1”对目标寄存器 R1 的位 1 清零，结果为 05H。

## 6.1.8.8 BITS — 位置 1 (Bit Set)

BITS dst.b

操作: dst(b) ← 1

BITS 指令将目的操作数的特定位设置为 1，而不影响其它的位。

标志位: 没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst   b   1	2	4	77	rb

**注释:** 在指令的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例:

假设: R1 = 07H:

BITS R1.3 → R1 = 0FH

工作寄存器 R1 的内容是 07H，指令“BITS R1.3”将目标寄存器 R1 的位 3 置为 1，结果为 0FH。

## 6.1.8.9 BOR — 位或 (Bit OR)

BOR dst, src.b

BOR dst.b, src

操作:  $dst(0) \leftarrow dst(0) \text{ OR } src(b)$   
或  
 $dst(b) \leftarrow dst(b) \text{ OR } src(0)$

源操作数（或者目的操作数）的特定位置与目的操作数（源操作数）的最低位进行逻辑或，运算结果保存在目的操作数的特定位置，目的操作数的其它位不受影响。源操作数不受影响。

标志位: **C:** 不受影响  
**Z:** 如果结果为 0，被置 1；否则，被清 0  
**S:** 总是被清零  
**V:** 不确定  
**D:** 不受影响  
**H:** 不受影响

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	dst   b   0	src	3	6	07	r0	Rb
opc	src   b   1	dst	3	6	07	Rb	r0

注释: 在 3 字节指令格式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例:

假设: R1 = 07H 和寄存器 01H = 03H:

```
BOR R1, 01H.1 → R1 = 07H, 寄存器 01H = 03H
BOR 01H.2, R1 → 寄存器 01H = 07H, R1 = 07H
```

在第一个例子里面，目的寄存器 R1 的内容为 07H，源寄存器 01H 的内容是 03H，指令“BOR R1, 01H.1”将寄存器 01H 的位 1 与 R1 的位 0 进行逻辑或运算，结果(还是 07H )保存在 R1 中。

在第二个例子里面，目的寄存器 01H 的内容为 03H，源寄存器 R1 的内容是 07H，指令“BOR 01H.2, R1”将 01H 的位 2 与寄存器 R1 的位 0 进行逻辑或运算，结果(07H)保存在寄存器 01H 中。

## 6.1.8.10 BTJRF — 位测试，若为假相对跳转 (Bit Test, Jump Relative on False)

BTJRF dst, src.b

**操作:** 如果目标操作数的位 b 为“0”，那么  $PC \leftarrow PC + dst$

测试源原操作数的特定位，如果为“0”，则将相对地址累加到程序计数器(PC)，并从新的 PC 地址开始执行程序；否则，执行 BTJRF 后面的指令。

**标志位:** 没有标志受影响。

**格式:**

(注释 1)			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	src   b   0	dst	3	10	37	RA	rb

**注释:** 在 3 字节指令各式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

**编程实例:**

假设: R1 = 07H:

BTJRF SKIP, R1.3 → PC 跳转到 SKIP 地址处

如果工作寄存器 R1 的内容是 07H，指令“BTJRF SKIP, R1.3”测试 R1 的位 3。因其为“0”，相对地址将累加到 PC，然后 PC 跳转到 SKIP 地址处。（记住地址跳转的范围必须在 +127 至 -128 之间）

### 6.1.8.11 BTJRT — 位测试，若为真，相对跳转 (Bit Test, Jump Relative on True)

**BTJRT**            dst, src.b

**操作:**            如果目标操作数的位 b 为“1”，那么  $PC \leftarrow PC + dst$

测试源操作数的特定位，如果为“1”，则将相对地址累加到程序计数器(PC)，并从新的 PC 地址开始执行程序；如果为“0”，执行 BTJRT 后面的指令。

**标志位:**        没有标志位受影响。

**格式:**

(注释 1)			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	src   b   1	dst	3	10	37	RA	rb

**注释:**    在 3 字节指令各式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

**编程实例:**

假设:            R1 = 07H:

```
BTJRT SKIP, R1.1
```

工作寄存器 R1 的内容是 07H，指令“BTJRT SKIP, R1.1”测试寄存器 R1 的位 1。因该位为“1”，相对地址将被叠加到 PC，并跳转到新的 PC 地址 SKIP 处。（记住跳转范围必须在 +127至 -128 之间）



## 6.1.8.12 BXOR — 位异或 (Bit XOR)

**BXOR**            dst, src.b

**BXOR**            dst.b, src

**操作:**            dst(0) ← dst(0) XOR src(b)  
或  
dst(b) ← dst(b) XOR src(0)

源操作数（或者目的操作数）的特定位置与目的操作数（源操作数）的最低位进行逻辑异或运算。结果保存在目的寄存器的特定位置中。其他位不受影响。源操作数不受影响。

**标志位:**        **C:**     不受影响  
                  **Z:**     如果结果为 0, 被置 1; 否则, 被清 0  
                  **S:**     总是被清零  
                  **V:**     不确定  
                  **D:**     不受影响  
                  **H:**     不受影响

**格式:**

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	dst   b   0	src	3	6	27	r0	Rb
opc	src   b   1	dst	3	6	27	Rb	r0

**注释:**     在 3 字节指令各式的第二个字节中, 目的操作数地址是 4 位, 位地址是 3 位。

**编程实例:**

假设: R1 = 07H (00000111B) 和寄存器 01H = 03H (00000011B):

```

BXOR R1,01H.1   →   R1 = 06H, 寄存器 01H = 03H
BXOR 01H.2,R1  →   寄存器 01H = 07H, R1 = 07H

```

在第一个例子中, 目的寄存器 R1 的内容是 07H, 源寄存器 01H 的内容是 03H, 指令“BXOR R1, 01H.1”将源寄存器 01H 的位 1 和 R1 的位 0 进行逻辑异或, 结果保存在 R1 的位 0, 将 R1 的值从 07H 改写为 06H。源寄存器 01H 的值不受影响。

## 6.1.8.13 CALL — 程序调用 (Call Procedure)

CALL dst

操作:

```

SP    ←    SP - 1
@SP   ←    PCL
SP    ←    SP - 1
@SP   ←    PCH
PC    ←    dst

```

PC 的当前内容被压入堆栈，也就是紧跟 CALL 指令之后的指令地址。

然后，指定的目标地址被送给PC，指向子程序的第一条指令地址。在子程序末尾，用返回指令 RET 返回到原来的程序流程，继续执行主程序。RET 指令的执行，将 PC 值从堆栈顶部弹出。

标志位: 没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	3	14	F6	DA
opc	dst	2	12	F4	IRR
opc	dst	2	14	D4	IA

编程实例:

假设: R0 = 35H, R1 = 21H, PC = 1A47H, 和 SP = 0002H:

```

CALL 3521H → SP = 0000H
              (存储器 0000H = 1AH, 0001H = 4AH,
              4AH 是指令后面的地址)
CALL @RR0 → SP = 0000H (0000H = 1AH, 0001H = 49H)
CALL #40H → SP = 0000H (0000H = 1AH, 0001H = 49H)

```

在第一个例子中，如果 PC 值为 1A47H，堆栈指针内容为 0002H，指令“CALL 3521H”将当前的PC数值压入堆栈顶，堆栈指针现在指向 0000H，然后 PC 装载入 3521H，从子程序的第一条指令地址开始顺序执行。

如果 PC 和 SP 的内容与第一个例子相同，指令“CALL @RP0”运行的结果基本相同，除了堆栈0001H的内容是 49H（因为是 2 字节指令），然后 PC 装载数值 3521H，并顺序执行指令。如果PC和堆栈指针的内容与第一个例子相同，如果程序抵制 0040H 的内容为 35H，41H 的内容为21H，指令“CALL #40H”产生和第二个例子相同的结果。

## 6.1.8.14 CCF — 进位标志位取反 (Complement Carry Flag)

## CCF

操作:  $C \leftarrow \text{NOT } C$

进位标志 C 取反。如果C=“1”，进位标志变成 0；如果C=“0”，进位标志变成 1。

标志位: C: 取反.

其他的标志位不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	EF

编程实例:

假设: 进位标志C=“0”:

CCF

如果进位标志 C=“0”，指令 CCF 对该标志为取反，在标志位寄存器中，该位从“0”变成“1”。

## 6.1.8.15 CLR — 清零 (Clear)

CLR           dst

操作:           dst ← "0"

目的操作数被清零。

标志位:        没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	B0	R
			4	B1	IR

编程实例:

假设:           寄存器 00H = 4FH, 寄存器 01H = 02H, 和寄存器 02H = 5EH:

CLR   00H   →    寄存器 00H = 00H

CLR   @01H →    寄存器 01H = 02H, 寄存器 02H = 00H

在寄存器寻址模式中, 指令“CLR 00H”把目的寄存器 00H 的内容清零。在第二个例子中, 指令“CLR @01H”使用间接寄存器寻址模式, 把寄存器 02H 清零。

## 6.1.8.16 COM — 取反 (complement)

COM           dst

操作:           dst ← NOT dst

对目标地址的内容取反，所有的“1”变成“0”，所有的“0”变成“1”。

标志位:

**C:**        不受影响  
**Z:**        如果结果为 0，被置 1；否则，被清 0  
**S:**        如果结果是负数，被置 1；否则，被清 0  
**V:**        总是被清 0  
**D:**        不受影响  
**H:**        不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst		
<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px 10px;">opc</td> <td style="padding: 2px 10px;">dst</td> </tr> </table>	opc	dst		2	4	60	R
	opc	dst					
			4	61	IR		

编程实例:

假设:           R1 = 07H 和寄存器 07H = 0F1H:

COM   R1     →     R1 = 0F8H

COM   @R1   →     R1 = 07H, 寄存器 07H = 0EH

在第一个例子中，目的寄存器 R1 的内容是 07H，指令“COM R1”对R1的所有位取反，所有的逻辑“1”变成逻辑“0”，所有的逻辑“0”变成逻辑“1”，结果为 0F8H。

在第二个例子中，应用间接寄存器寻址模式对目的寄存器 07H 的内容取反，结果为 0EH。

## 6.1.8.17 CP — 比较 (Compare)

CP dst, src

操作: dst – src

源操作数和目的操作数作比较(相减), 根据结果设置适当的标志位。  
源操作数和目的操作数均不受影响。

标志位: **C:** 如果源操作数大于目的操作数, 被置 1; 否则, 被清 0  
**Z:** 如果结果为 0, 被置 1; 否则, 被清 0  
**S:** 如果结果是负数, 被置 1; 否则, 被清 0  
**V:** 总是被清零  
**D:** 不受影响  
**H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式	
					dst	src
opc	dst   src	2	4	A2	r	r
			6	A3	r	lr
opc	src   dst	3	6	A4	R	R
			6	A5	R	IR
opc	dst   src	3	6	A6	R	IM

编程实例:

1. 假设: R1 = 02H 和 R2 = 03H:

CP R1, R2 → 对 C 和 S 标志位置 1

目的寄存器 R1 的内容为 02H, 源寄存器 R2 的内容为 03H, 指令“CP R1, R2”把 R1 减去 R2, 因为产生了借位, 结果是负数, 所以 C 和 S 被置 1。

2. 假设: R1 = 05H 和 R2 = 0AH:

```
CP R1, R2
JP UGE, SKIP
INC R1
SKIP LD R3, R1
```

在这个例子中, 目的寄存器 R1 的内容是 05H, 小于源寄存器 R2 的内容 0AH。指令“CP R1, R2”使得 C = “1”, JP 指令没有跳转到 SKIP 处。当执行完指令“LD R3, R1”, 寄存器 R3 的内容为 06H。

### 6.1.8.18 CPIJE — 比较，增加一，若相等跳转 (Compare, Increment, and Jump on Equal)

**CPIJE**            dst, src, RA

**操作:**            If  $dst - src = "0"$ ,  $PC \leftarrow PC + RA$   
 $lr \leftarrow lr + 1$

源操作数与目的操作数作比较 (相减)。若结果为“0”，则相对地址被叠加到 PC 上，从 PC 指向的新地址开始执行程序。否则，继续执行 CPIJE 后面的指令。  
 无论哪种情况下，在执行下一条指令前，源指针增加 1。

**标志位:**        没有标志位受影响

**格式:**

				字节数	时钟周期	指令代码 (Hex)	寻址模式	
							dst	src
opc	src	dst	RA	3	12	C2	r	lr

**注释:**        执行时间是 18 个时钟周期 (跳转) 或者 16 个时钟周期 (无跳转)

**编程实例:**

假设:            R1 = 02H, R2 = 03H, 和寄存器 03H = 02H:

CPIJE R1,@R2,SKIP →     R2 = 04H, PC 跳转到地址 SKIP 处

在这个例子中，工作寄存器 R1 的内容是 02H，工作寄存器 R2 的内容是 03H，寄存器 03H 的内容是 02H，指令“CPIJE R1,@R2, SKIP”比较@ R2 的内容 02H 和 02H，因为比较的结果是相等的，跳转到新的 PC 地址 SKIP 处。源寄存器 R2 加 1 变成 04H。

记住 CPIJE 指令跳转的地址范围须介于 +127 至 -128 之间。

### 6.1.8.19 CPIJNE — 比较，增加一，不等跳转 (Compare, Increment, Jump on Non-Equal)

**CPIJNE**            dst, src, RA

**操作:**            If dst – src = "0", PC ← PC + RA  
                       lr ← lr + 1

源操作数与目的操作数作比较(相减)，如果结果不为“0”，则将相对地址叠加到 PC 上，从 PC 指向的新地址开始执行程序。否则，继续执行 CPIJE 后面的指令。  
 无论在哪种情况下，在执行下一条指令前，源指针增加 1。

**标志位:**            没有标志位受影响

**格式:**

				字节数	时钟周期	指令代码 (Hex)	寻址模式	
							dst	src
opc	src	dst	RA	3	12	D2	r	lr

**注释:**    执行时间是 18 个时钟周期（跳转）或者 16 个时钟周期（无跳转）

**编程实例:**

假设:            R1 = 02H, R2 = 03H, 和寄存器 03H = 04H:

CPIJNE            R1,@R2,SKIP →        R2 = 04H, PC 跳转到地址 SKIP

在这个例子中，工作寄存器 R1 的内容是 02H，工作寄存器 R2 的内容是 03H，寄存器 03H 的内容是 04H，指令“CPIJE R1, @R2, SKIP”比较 @R2 的内容 04H 和 02H，因为比较的结果是不相等的，跳转到新的 PC 地址 SKIP 处。源寄存器 R2 加 1 变成 04H。

记住 CPIJNE 指令跳转的地址范围须介于 +127 至 -128 之间。



## 6.1.8.20 DA — 十进制调整 (Decimal Adjust)

DA dst

操作: dst ← DA dst

在加法或者减法运算后，将结果调整为 2 个 4 位 BCD 码。  
对于加法 (ADD, ADC) 或者减法 (SUB, SBC) 下面的表格指明了操作如何进行。

指令	DA前的 进位标志	位7-4 (十六进制)	DA前H标志	位3-0 (十六进制)	要增加的数字	DA后的 进位标志
	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
ADD	0	A-F	0	0-9	60	1
ADC	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
	0	0-9	0	0-9	00 = -00	0
SUB	0	0-8	1	6-F	FA = -06	0
SBC	1	7-F	0	0-9	A0 = -60	1
	1	6-F	1	6-F	9A = -66	1

标志位:

- C:** 如果发生进位，被置 1；否则，被清 0
- Z:** 如果结果为 0，被置 1；否则，被清 0
- S:** 如果结果是负数，被置 1；否则，被清 0
- V:** 没有定义
- D:** 不受影响
- H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	40	R
			4	41	IR

DA (续)

## 编程实例:

假设: 工作寄存器R0的内容是15 (BCD), 工作寄存器 R1 的内容是 27 (BCD), 地址 27H 的内容是 46 (BCD):

```
ADD  R1,R0 ;      C ← "0", H ← "0", Bits 4-7 = 3, bits 0-3 = C, R1 ← 3CH
DA   R1      ;      R1 ← 3CH + 06
```

如果用 BCD 数值 15 和 27 进行加法运算, 结果是 27。然而结果是不对的, 当加法的时候, 使用的是标准的二进制运算:

```

      0 0 0 1   0 1 0 1 15
+     0 0 1 0   0 1 1 1 27
-----
      0 0 1 1   1 1 0 0 =   3CH
```

DA 指令调整了运算结果, 进而得到正确的结果:

```

      0 0 1 1   1 1 0 0
+     0 0 0 0   0 1 1 0
-----
      0 1 0 0   0 0 1 0 =   42
```

假设同样的数值, 指令:

```
SUB  27H,R0      ;      C ← "0", H ← "0", Bits 4-7 = 3, bits 0-3 = 1
DA   @R1         ;      @R1 ← 31-0
```

执行的结果为 31 (BCD), 保存在地址 27H (@R1)。

## 6.1.8.21 DEC — 字节减 1 (Decrement)

DEC dst

操作:  $\text{dst} \leftarrow \text{dst} - 1$ 

目的操作数的内容减 1

标志位:

**C:** 不受影响  
**Z:** 如果结果为 0, 被置 1; 否则, 被清 0  
**S:** 如果结果是负数, 被置 1; 否则, 被清 0  
**V:** 如果结果溢出, 被置 1; 否则, 被清 0  
**D:** 不受影响  
**H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst		
<table border="1" style="margin: auto;"> <tr> <td style="padding: 5px;">opc</td> <td style="padding: 5px;">dst</td> </tr> </table>	opc	dst		2	4	00	R
	opc	dst					
			4	01	IR		

编程实例:

假设: R1 = 03H 和寄存器 03H = 10H:

DEC R1 → R1 = 02H  
 DEC @R1 → 寄存器 03H = 0FH

在第一个例子中, 如果工作寄存器 R1 的内容是 03H, 值令 “DEC R1” 将该 16 进制数减 1, 结果为 02H。  
 在第二个例子中, “DEC @R1” 将数值 10H 减 1, 得到 0FH, 保存在地址 03H。

## 6.1.8.22 DECW — 字减 1 (Decrement Word)

DECW           dst

操作:           dst ← dst - 1

目的地址(须为偶数)的内容减 1, 把操作数视为 16 位数据。

标志位:       **C:**     不受影响  
               **Z:**     如果结果为 0, 被置 1; 否则, 被清 0  
               **S:**     如果结果是负数, 被置 1; 否则, 被清 0  
               **V:**     如果有溢出, 被置 1; 否则, 被清 0  
               **D:**     不受影响  
               **H:**     不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	8	80	RR
			8	81	IR

编程实例:

假设: R0 = 12H, R1 = 34H, R2 = 30H, 寄存器 30H = 0FH, 和寄存器 31H = 21H:

```
DECW RR0  →  R0 = 12H, R1 = 33H
DECW @R2  →  寄存器 30H = 0FH, 寄存器 31H = 20H
```

在第一个例子中, 目的寄存器 R0 内容是 12H, 寄存器 R1 内容 34H, 指令“DECW RR0”把R1, R0 当作 16 位数, 减 1 以后, 得到的结果为 33H。

**注释:** 当 DECW 指令与 Z 标志一起使用时, 可能导致系统错误。为避免这个问题, 推荐按照如下方法来使用 DECW 指令:

```
LOOP:  DECW  RR0
        LD   R2,R1
        OR   R2,R0
        JR   NZ,LOOP
```

### 6.1.8.23 DI — 屏蔽全局中断 (Disable Interrupts)

#### DI

操作:           SYM(0) ← 0

系统模式控制寄存器的位 0, SYM.0 被清零, 将禁止全局中断。  
中断请求仍然会置起相应的中断悬挂标志, 但 CPU 不会响应中断服务程序,  
因为中断处理被屏蔽了。

标志位:        没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	8F

编程实例:

假设:        SYM = 01H:

DI

如果 SYM 寄存器的值是 01H, 指令 DI 将清零 SYM.0, 屏蔽所有的中断处理。

在改变 IMR、中断标志位、中断源控制寄存器之前, 确保 DI 状态。

## 6.1.8.24 DIV — 无符号除法 (Unsigned Divide)

**DIV**                   dst, src

**操作:**                   dst ÷ src  
                   dst (UPPER) ← REMAINDER  
                   dst (LOWER) ← QUOTIENT

目的操作数 (16 位) 除以源操作数 (8 位), 商 (8 位) 保存在目的操作数的低字节, 余数 (8 位) 则保存在目的操作数的高字节。如果商  $\geq 28$ , 保存在目的操作数高低字节的商和余数是不正确的。所有的操作数都是无符号整数。

**标志位:**               **C:**        如果 V 标志被设置并且商在  $2^8$  和  $2^9 - 1$  范围之间, 被置 1; 否则, 被清 0。  
                   **Z:**        如果除数或者商 = “0”, 被置 1; 否则, 被清 0。  
                   **S:**        如果商的最高位 = “1”, 被置 1; 否则, 被清 0。  
                   **V:**        如果商  $\geq 2^8$  或者除数 = “0”, 被置 1; 否则, 被清 0。  
                   **D:**        不受影响  
                   **H:**        不受影响。

**格式:**

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	src	dst	3	26/10*	94	RR	R
				26/10*	95	RR	IR
				26/10*	96	RR	IM

**注释:**    如果除数为 0, 运行时间需要 10 个时钟周期; 其他情况下, 执行时间为 26 个时钟周期。

**编程实例:**

假设:           R0 = 10H, R1 = 03H, R2 = 40H, 寄存器 40H = 80H:

```

DIV  RR0, R2      →   R0 = 03H, R1 = 40H
DIV  RR0, @R2    →   R0 = 03H, R1 = 20H
DIV  RR0, #20H   →   R0 = 03H, R1 = 80H

```

在第一个例子中, 目的寄存器 RR0 的内容是 10H (R0) 和 03H (R1), 寄存器 R2 的内容是 40H。指令 “DIV RR0, R2” 将 16 位 RR0 被除数除以 8 位除数 R2, 除法运算后, R0 内容为 03H, R1 的内容为 40H。8 位余数保存在目的寄存器 RR0 的上半部分 (R0), 商则保存在下半部分 (R1)。

## 6.1.8.25 DJNZ — 减 1, 如果非零, 跳转 (Decrement and Jump if Non-Zero)

**DJNZ**            r, dst

**操作:**             $r \leftarrow r - 1$   
                     If  $r \neq 0, PC \leftarrow PC + dst$

作为计数器的工作寄存器值首先减 1, 如果结果不为“0”, 相对地址被累加到 PC, 然后程序跳转到新的地址执行。相对地址的范围是 +127 至 -128, PC 的初始值是紧跟在 DJNZ 指令后面的指令地址。

**注释:**            DJNZ 指令的时候, 被用来做计数器的工作寄存器必须通过 SRP, SRP0, SRP1 指令设置为 0C0H~0C1H 中的一个。标志位: 没有标志位受影响

**格式:**

			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst		
r		opc		dst	2	8 (jump taken)	rA	RA
						8 (no jump)	r = 0 to F	

**编程实例:**

假设: R1 = 02H 和 LOOP 是相对地址的地址标志符:

```

LOOP  SRP  #0C0H
      DJNZ  R1, LOOP

```

指令 DJNZ 经常被用在循环程序中。大多数情况下, 用地址标号而非数字相对地址来做目的操作数。在这个例子中, 工作寄存器 R1 内容是 02H, LOOP 是相对地址的标号。

指令“DJNZ R1, LOOP”首先将寄存器 R1 减 1, 得到结果 01H, 因为减 1 后 R1 的内容非“0”, 程序将跳转到标号 LOOP 指向的地址执行。

## 6.1.8.26 EI — 使能全局中断 (Enable Interrupts)

## EI

操作:           SYM (0) ← 1

EI 指令对系统模式寄存器(SYM)的最低位 SYM.0 置 1, 这将允许响应中断服务程序(假定该中断具有最高优先级)。如果中断处理被屏蔽(通过执行 DI)时, 又发生中断, 可通过执行EI指令, 来执行中断服务程序。

标志位:        没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)
Opc	1	4	9F

编程实例:

假设:        SYM = 00H:

EI

如果 SYM 寄存器的内容是 00H, 也就是说, 全局中断被屏蔽, 执行指令“EI” 将把SYM寄存器设置为 01H, 使能所有中断 (SYM.0 是全局中断的使能位)。



6.1.8.27 ENTER — 进入 (Enter)

ENTER

操作:            SP     ←     SP - 2  
                  @SP ←     IP  
                  IP     ←     PC  
                  PC     ←     @IP  
                  IP     ←     IP + 2

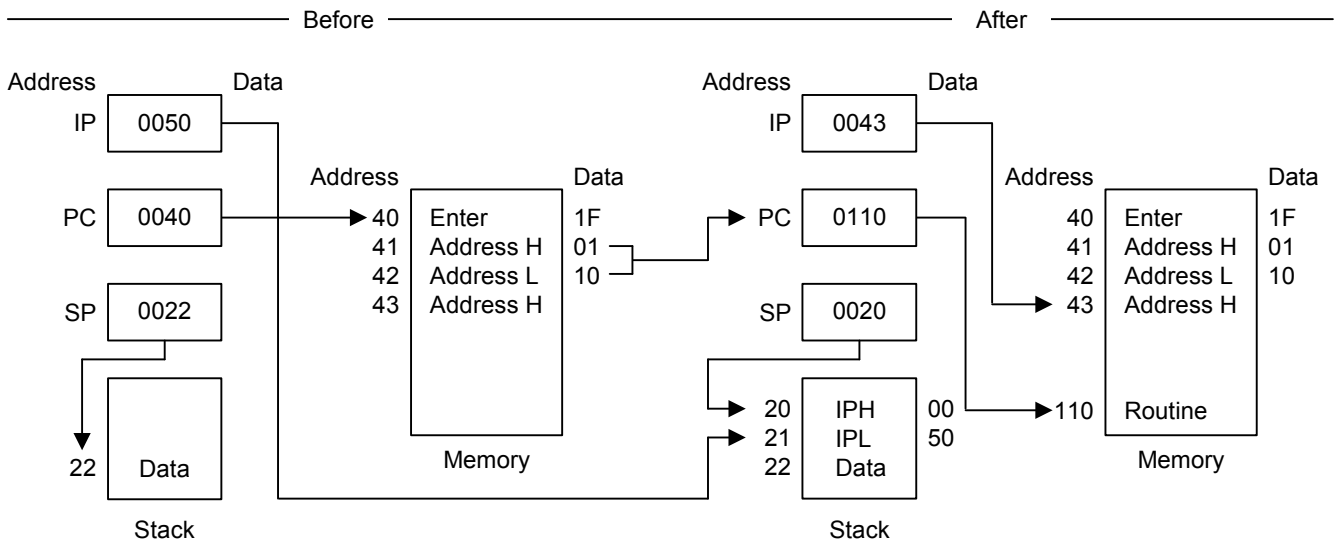
这条指令在执行线性代码语言时非常有用。指令指针的内容压入堆栈，然后把程序计数器（PC）的值写入指令指针，程序存储器中指令指针所指向的字数据载入 PC，并且指令指针的值增加 2。

标志位:            没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	14	1F

编程实例:            下图给出了一个如何使用 ENTER 指令的例子。



6.1.8.28 EXIT — 退出 (Exit)

EXIT

操作: IP ← @SP  
 SP ← SP + 2  
 PC ← @IP  
 IP ← IP + 2

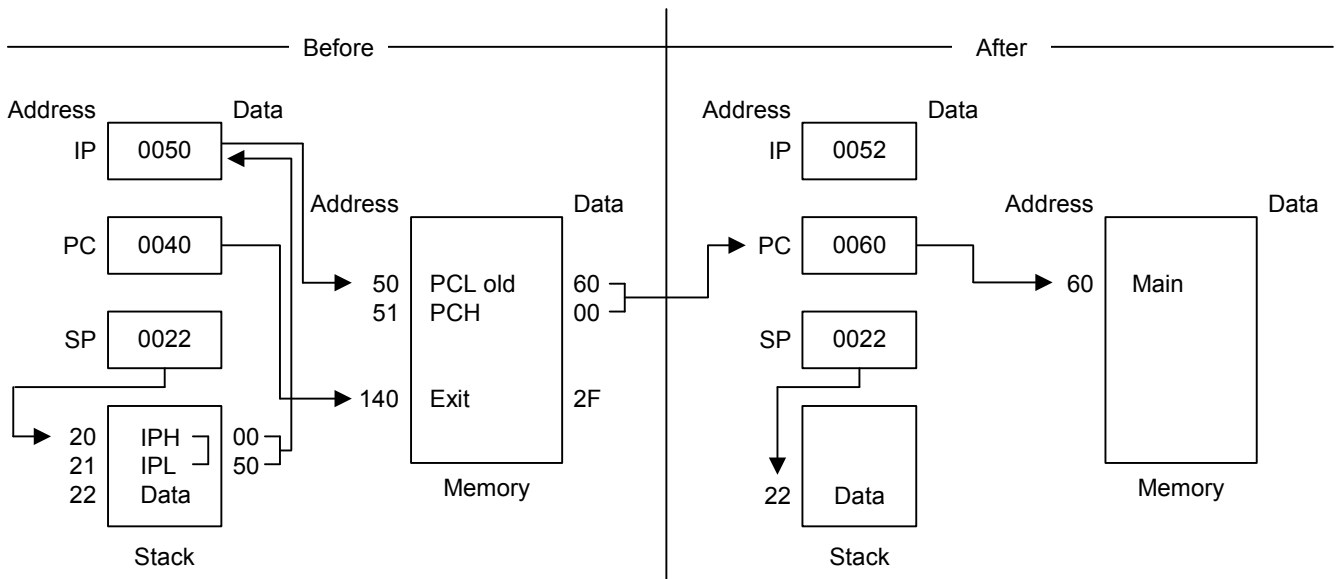
这条指令在执行线性代码语言时非常有用。被压入堆栈的值弹出并载入指令指针，程序存储器中被指令指针指向的字数据载入程序计数器，并且指令指针的值增加 2。

标志位: 没有标志位受影响

格式:

字节数	时钟周期	指令代码 (Hex)
1	14 (内部堆栈) 16 (外部堆栈)	2F

编程实例: 下图给出了一个如何使用 EXIT 指令的例子。



## 6.1.8.29 IDLE — 空闲指令 (Idle Operation)

## IDLE

操作: (见描述)

IDLE 指令将停止 CPU 时钟但允许系统时钟继续工作。IDLE 模式可以被中断请求 (IRQ) 或者是外部复位操作唤醒。在应用程序中, IDLE 指令后必须立即执行至少 3 个 NOP 指令。这是为了保证在下一条指令执行之前, 系统时钟有足够的时间间隔来稳定时钟信号。如果在 IDLE 指令后没有 3 个或多个的 NOP 指令, 内部总线的悬浮状态将导致漏电流的产生。

标志位: 没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	1	4	6F	-	-

编程实例:

指令	
IDLE	; 停止 CPU 时钟但不停系统时钟
NOP	
NOP	
NOP	

## 6.1.8.30 INC — 加 1 (Increment)

INC dst

操作:  $dst \leftarrow dst + 1$ 

目标操作数的内容增加1

标志位:

**C:** 没有影响  
**Z:** 如果结果为“0”则置 1; 否则清零  
**S:** 如果结果为负数则置 1; 否则清零  
**V:** 如果发生溢出则置 1; 否则清零  
**D:** 没有影响  
**H:** 没有影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
dst   opc	1	4	rE r = 0 到 F	r
opc   dst	2	4	20	R
		4	21	IR

编程实例:

假如: R0 = 1BH, 寄存器 00H = 0CH, 寄存器 1BH = 0FH:

```
INC R0    → R0 = 1CH
INC 00H   → 寄存器 00H = 0DH
INC @R0   → R0 = 1BH, 寄存器 01H = 10H
```

在第一个例子中, 如果目标工作寄存器R0的值为1BH, 那么语句“INC R0”将 R0 的值变为1CH。

第二个例子演示了 INC 指令对寄存器 00H 产生的效果, 假定寄存器的值为 0CH。

第三个例子中, INC 指令用在间接寄存器 (IR) 寻址模式中, 将寄存器 1BH 的值由 0FH 变为10H。

## 6.1.8.31 INCW — 字加 1 (Increment Word)

INCW            dst

操作:             $dst \leftarrow dst + 1$ 

目标操作数（必须是偶地址）中的一个字节和下一地址中的字节数据合在一起作为一个 16 位数据，整个 16 位的数据增加 1

标志位:        **C:**     没有影响  
                  **Z:**     如果结果为“0”则置 1；否则清零  
                  **S:**     如果结果为负数则置 1；否则清零  
                  **V:**     如果发生溢出则置 1；否则清零  
                  **D:**     没有影响  
                  **H:**     没有影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	8	A0	RR
			8	A1	IR

编程实例:

假如: R0 = 1AH, R1 = 02H, 寄存器 02H = 0FH, 寄存器 03H = 0FFH:

```
INCW RR0  →  R0 = 1AH, R1 = 03H
INCW @R1  →  寄存器 02H = 10H, 寄存器 03H = 00H
```

在第一个例子中，工作寄存器对 RR0 的内容是 1AH(R0) 和 02H(R1)。语句“INCW RR0”将 16 位的目标数据加 1，使寄存器 R1 的值变为 03H。第二个例子中，语句“INCW @R1”用间接寄存器（IR）寻址模式将寄存器 03H 的值从 0FFH 变为 00H，寄存器 02H 的值从 0FH 变为 10H。

**注释:** 如果 Zero (Z) 标志位（标志位.6）与 INCW 指令一起使用，可能会发生冲突而导致系统错误。为了避免这个问题，建议按下面的方法使用 INCW 指令：

```
LOOP:  INCW  RR0
        LD   R2,R1
        OR   R2,R0
        JR   NZ,LOOP
```

## 6.1.8.32 IRET — 中断返回 (Interrupt Return)

<b>IRET</b>	IRET (正常)	IRET (快速)
<b>操作:</b>	标志位 $\leftarrow$ @SP SP $\leftarrow$ SP + 1 PC $\leftarrow$ @SP SP $\leftarrow$ SP + 2 SYM(0) $\leftarrow$ 1	PC $\leftrightarrow$ IP 标志位 $\leftarrow$ 标志位' FIS $\leftarrow$ 0

该指令用在中断服务程序的末尾。该指令将恢复标志寄存器和程序计数器的值，同时重新使能全局中断。只有在快速中断状态位 (FIS, 标志位寄存器位 1, 0D5H) 被清零时 (= “0”)，才执行“正常IRET”。快速中断产生时，对于在中断服务程序开始时被置 1 的 FIS 位，IRET 会将其清零。

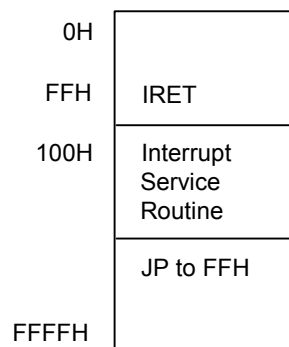
**标志位:** 所有标志位恢复到初始状态 (即中断发生以前的状态)

**格式:**

IRET (正常)	字节数	时钟周期	指令代码 (Hex)
Opc	1	10 (内部堆栈) 12 (外部堆栈)	BF
IRET(快速)	字节数	时钟周期	指令代码 (Hex)
opc	1	6	BF

**编程实例:** 在下图中，中断使能之前，在主程序中将指令指针初始化为 100H。中断发生时，程序计数器和指令指针的内容相交换，这使得 PC 跳转到地址 100H 而 IP 则保存返回地址。中断服务程序的最后一条指令通常是跳转到地址 FFH 处的 IRET。

这将使指令指针“重新”被赋值为 100H，且程序计数器跳回到主程序。现在，下一个中断可以发生了，IP 寄存器的值仍然为 100H。



**注释:** 上面快速中断的例子中，如果最后的指令不是跳转到 IRET，那么必须注意最后两条指令的顺序。在 IRET 指令之后，不可紧跟用于中断状态清除的指令 (例如 IPR 寄存器清 0)。

## 6.1.8.33 JP — 跳转 (Jump)

**JP**            cc,dst    (条件跳转)

**JP**            dst        (无条件跳转)

**操作:**        如果 cc 为真, PC ← dst

如果转移条件为真, 那么条件跳转指令把系统控制交给目标地址; 否则执行紧跟 JP 指令之后的指令。无条件跳转指令只是简单的用目标地址替换 PC 的内容, 然后程序控制交给由 PC 指定的语句。

**标志位:**     没有标志位受影响

**格式:** (1)

(2)		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
cc	opc	3	8	ccD cc = 0 to F	DA
opc	dst	2	8	30	IRR

**注释:**

1. 3 字节格式用于条件跳转, 2 字节格式用于无条件跳转
2. 在 3 字节指令格式 (条件跳转) 的第一字节中, 条件码和指令代码各占 4 位

**编程实例:**

假如: 进位标志 (C) = “1”, 寄存器 00 = 01H, 寄存器 01 = 20H:

```
JP    C, LABEL_W  →    LABEL_W = 1000H, PC = 1000H
JP    @00H        →    PC = 0120H
```

第一个例子是条件跳转 JP。假定进位标志为 “1”, 指令 “JP C, LABEL\_W” 将 PC 的值替换为 1000H 并且跳转到该地址。如果标志位没有被置 1, 那么程序将立即转到紧跟 JP 后的那条指令。

第二个例子为无条件跳转 JP。指令 “JP @00” 将 PC 的内容替换为寄存器对 00H 和 01H 的值, 亦即 0120H。

### 6.1.8.34 JR — 相对跳转指令 (Jump Relative)

**JR**                    cc, dst

**操作:**                如果 cc 为真,  $PC \leftarrow PC + dst$

如果转移条件为真, 那么程序计数器值加上相对地址, 并把程序控制转到该地址处的指令; 否则执行紧跟 JR 指令后的那条指令 (见本章: 转移条件码列表)。

相对地址的范围是  $-128 \sim +127$ , 并且程序计数器的初值被认为是紧跟JR指令之后的第一条指令地址。

**标志位:**             没有标志位受影响

**格式:**

(1)			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
cc	opc	dst	2	6	ccB	RA
cc = 0 到 F						

**注释:**    2 字节指令格式中的第一个字节中, 条件码和指令代码各占 4 位

**编程实例:**

假如:                进位标志位 = “1” 并且 LABEL\_X = 1FF7H:

JR     C, LABEL\_X    →     PC = 1FF7H

如果进位标志位为 “1” (也就是, 转移条件为真), 指令 “JR C, LABEL\_X” 会将程序控制交给当前 PC 指向的地址; 否则, 执行紧跟 JR 后的程序指令。



## 6.1.8.35 LD — 传送数据 (Load)

LD dst, src

操作: dst ← src

源操作数的内容将赋给目标操作数。源操作数的内容不受影响。

标志位: 没有标志位受影响

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
dst   opc	src		2	4	rC	r	IM
				4	r8	r	R
src   opc	dst		2	4	r9	R	r
opc	dst   src		2	4	C7	r	lr
				4	D7	lr	r
opc	src	dst	3	6	E4	R	R
				6	E5	R	IR
opc	dst	src	3	6	E6	R	IM
				6	D6	IR	IM
opc	src	dst	3	6	F5	IR	R
opc	dst   src	x	3	6	87	r	x [r]
opc	src   dst	x	3	6	97	x [r]	r

LD (续)

编程实例:

假如: R0 = 01H, R1 = 0AH, 寄存器 00H = 01H, 寄存器 01H = 20H,  
寄存器 02H = 02H, LOOP = 30H, 寄存器 3AH = 0FFH:

LD	R0,#10H	→	R0 = 10H
LD	R0,01H	→	R0 = 20H, 寄存器 01H = 20H
LD	01H,R0	→	寄存器 01H = 01H, R0 = 01H
LD	R1,@R0	→	R1 = 20H, R0 = 01H
LD	@R0,R1	→	R0 = 01H, R1 = 0AH, 寄存器 01H = 0AH
LD	00H,01H	→	寄存器 00H = 20H, 寄存器 01H = 20H
LD	02H,@00H	→	寄存器 02H = 20H, 寄存器 00H = 01H
LD	00H,#0AH	→	寄存器 00H = 0AH
LD	@00H,#10H	→	寄存器 00H = 01H, 寄存器 01H = 10H
LD	@00H,02H	→	寄存器 00H = 01H, 寄存器 01H = 02, 寄存器 02H = 02H
LD	R0,#LOOP[R1]	→	R0 = 0FFH, R1 = 0AH
LD	#LOOP[R0],R1	→	寄存器 31H = 0AH, R0 = 01H, R1 = 0AH

## 6.1.8.36 LDB — 传送位数据 (Load Bit)

LDB dst, src.b

LDB dst.b, src

操作: dst(0) ← src(b)  
或  
dst(b) ← src(0)

源操作数的指定位载入目标操作数的最低位，或者源操作数的最低位载入目标操作数的指定位。目标操作数的其它位都不受影响，源操作数也不受影响。

标志位: 没有标志位受影响

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	dst   b   0	src	3	6	47	r0	Rb
opc	src   b   1	dst	3	6	47	Rb	r0

注释: 指令格式的第二个字节中，目标（或源）地址占 4 位，位地址“b”占 3 位，LSB 地址值占 1 位

编程实例:

假如: R0 = 06H，通用寄存器 00H = 05H:

```
LDB R0,00H.2 → R0 = 07H, 寄存器 00H = 05H
LDB 00H.0,R0 → R0 = 06H, 寄存器 00H = 04H
```

第一个例子中，目标工作寄存器 R0 的值为 06H，源寄存器 00H 的值为 05H。指令“R0,00h.2”将寄存器 00H 中的第二位 (bit 2) 载入寄存器 R0 的最低位，R0 的值变为 07H。

第二个例子中，目标寄存器是 00H。指令“LD 00H.0, R0”将工作寄存器 R0 的最低位载入目标寄存器 00H 的指定位 (bit 0)，寄存器 00H 的值变为 04H。

## 6.1.8.37 LDC/LDE — 传送程序/外部数据存储器数据 (Load Memory)

LDC/LDE dst, src

操作: dst ← src

这条指令从程序或外部数据存储器中装载一个字节数据到工作寄存器，或者相反。源操作数不受影响。指令 LDC 用作程序存储器，LDE 用作外部数据存储器。对于程序存储器，编译器将“lrr”或“rr”的值编译成偶地址，而对于外部数据存储器则编译成奇地址。

标志位: 没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式					
				dst	src				
1.	<table border="1"><tr><td>opc</td><td>dst   src</td></tr></table>	opc	dst   src	2	10	C3	r lrr		
opc	dst   src								
2.	<table border="1"><tr><td>opc</td><td>src   dst</td></tr></table>	opc	src   dst	2	10	D3	lrr r		
opc	src   dst								
3.	<table border="1"><tr><td>opc</td><td>dst   src</td><td>XS</td></tr></table>	opc	dst   src	XS	3	12	E7	r XS [rr]	
opc	dst   src	XS							
4.	<table border="1"><tr><td>opc</td><td>src   dst</td><td>XS</td></tr></table>	opc	src   dst	XS	3	12	F7	XS [rr] r	
opc	src   dst	XS							
5.	<table border="1"><tr><td>opc</td><td>dst   src</td><td>XL<sub>L</sub></td><td>XL<sub>H</sub></td></tr></table>	opc	dst   src	XL <sub>L</sub>	XL <sub>H</sub>	4	14	A7	r XL [rr]
opc	dst   src	XL <sub>L</sub>	XL <sub>H</sub>						
6.	<table border="1"><tr><td>opc</td><td>src   dst</td><td>XL<sub>L</sub></td><td>XL<sub>H</sub></td></tr></table>	opc	src   dst	XL <sub>L</sub>	XL <sub>H</sub>	4	14	B7	XL [rr] r
opc	src   dst	XL <sub>L</sub>	XL <sub>H</sub>						
7.	<table border="1"><tr><td>opc</td><td>dst   0000</td><td>DA<sub>L</sub></td><td>DA<sub>H</sub></td></tr></table>	opc	dst   0000	DA <sub>L</sub>	DA <sub>H</sub>	4	14	A7	r DA
opc	dst   0000	DA <sub>L</sub>	DA <sub>H</sub>						
8.	<table border="1"><tr><td>opc</td><td>src   0000</td><td>DA<sub>L</sub></td><td>DA<sub>H</sub></td></tr></table>	opc	src   0000	DA <sub>L</sub>	DA <sub>H</sub>	4	14	B7	DA r
opc	src   0000	DA <sub>L</sub>	DA <sub>H</sub>						
9.	<table border="1"><tr><td>opc</td><td>dst   0001</td><td>DA<sub>L</sub></td><td>DA<sub>H</sub></td></tr></table>	opc	dst   0001	DA <sub>L</sub>	DA <sub>H</sub>	4	14	A7	r DA
opc	dst   0001	DA <sub>L</sub>	DA <sub>H</sub>						
10.	<table border="1"><tr><td>opc</td><td>src   0001</td><td>DA<sub>L</sub></td><td>DA<sub>H</sub></td></tr></table>	opc	src   0001	DA <sub>L</sub>	DA <sub>H</sub>	4	14	B7	DA r
opc	src   0001	DA <sub>L</sub>	DA <sub>H</sub>						

注释:

- 格式 5 和 6 的源操作数 [src] 或工作寄存器对 [rr] 不能使用工作寄存器对 0-1
- 格式 3 和 4 的目标地址“XS[rr]”和源地址“XS[rr]”均为一个字节
- 格式 3 和 4 的目标地址“XL[rr]”和源地址“XL[rr]”均为两个字节
- 格式 7 和 8 的 DA 和 r 源操作数值用于访问程序存储器，格式 9 和 10 则用于访问外部数据存储器
- LDE 指令可用于读/写 64K 字节的外部数据存储器。

## LDC/LDE (续)

## 编程实例:

假如: R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H; 程序存储器空间中  
0103H = 4FH, 0104H = 1A, 0105H = 6DH, 1104H = 88H. 外部数据存储空间中  
0103H = 5FH, 0104H = 2AH, 0105H = 7DH, 1104H = 98H:

LDC	R0,@RR2	;	R0 ← 程序存储器地址 0104H 的内容
		;	R0 = 1AH, R2 = 01H, R3 = 04H
LDE	R0,@RR2	;	R0 ← 外部数据存储器地址 0104H 的内容
		;	R0 = 2AH, R2 = 01H, R3 = 04H
LDC (注释)	@RR2,R0	;	11H (R0 的内容) 载入程序存储器地址 0104H (RR2)
		;	工作寄存器 R0, R2, R3 → 没有变化
LDE	@RR2,R0	;	11H (R0 的内容) 载入外部数据存储器地址 0104H (RR2)
		;	工作寄存器 R0, R2, R3 → 没有变化
LDC	R0,#01H[RR2]	;	R0 ← 程序存储器地址 0105H 的内容
		;	(01H + RR2),
		;	R0 = 6DH, R2 = 01H, R3 = 04H
LDE	R0,#01H[RR2]	;	R0 ← 外部数据存储器地址 0105H 的内容
		;	(01H + RR2), R0 = 7DH, R2 = 01H, R3 = 04H
LDC (注释)	#01H[RR2],R0	;	11H (R0 的内容) 载入程序存储器地址
		;	0105H (01H + 0104H)
LDE	#01H[RR2],R0	;	11H (R0 的内容) 载入外部数据存储器地址
		;	0105H (01H + 0104H)
LDC	R0,#1000H[RR2]	;	R0 ← 程序存储器地址 1104H 的内容
		;	(1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H
LDE	R0,#1000H[RR2]	;	R0 ← 外部数据存储器地址 1104H 的内容
		;	(1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H
LDC	R0,1104H	;	R0 ← 程序存储器地址 1104H 的内容, R0 = 88H
LDE	R0,1104H	;	R0 ← 外部数据存储器地址 1104H 的内容
		;	R0 = 98H
LDC (注释)	1105H,R0	;	11H (R0 的内容) 载入程序存储器地址
		;	1105H, (1105H) ← 11H
LDE	1105H,R0	;	11H (R0 的内容) 载入外部数据存储器地址
		;	1105H, (1105H) ← 11H

**注释:** 掩膜 ROM 类型的器件不支持 LDC/LDE 指令。

## 6.1.8.38 LDCD/LDED — 传送数据之后地址减 1 (Load Memory and Decrement)

LDCD/LDED dst, src

操作:           dst ← src  
                  rr ← rr - 1

该指令用于用户栈或在程序/数据存储器与寄存器卷之间批量传送数据。存储器的地址由工作寄存器对指定。源地址的内容载入目标地址，然后存储器地址自动减1，源操作数的内容不变。

LDCD 对应程序存储器，而 LDED 对应外部数据存储器。对于程序存储器，编译器将“lrr”或“rr”的值编译成偶地址，对于数据存储器则编译成奇地址。

标志位:           没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	dst   src	2	10	E2	r	lrr

编程实例:

假如:           R6 = 10H, R7 = 33H, R8 = 12H, 程序存储器地址 1033H = 0CDH,  
 外部数据存储器地址 1033H = 0DDH:

```

LDCD R8,@RR6    ;    0CDH (程序存储器 1033H 的内容) 载入R8
                 ;    RR6 减1
                 ;    R8 = 0CDH, R6 = 10H, R7 = 32H (RR6 ← RR6 - 1)
LDED  R8,@RR6    ;    0DDH (数据存储器地址 1033H 的内容) 载入R8
                 ;    RR6 减1 (RR6 ← RR6 - 1)
                 ;    R8 = 0DDH, R6 = 10H, R7 = 32H

```

## 6.1.8.39 LDCI/LDEI — 传送数据后地址加 1 (Load Memory and Increment)

LDCI/LDEI dst, src

操作:  $dst \leftarrow src$   
 $rr \leftarrow rr + 1$

该指令用于用户栈或在程序/数据存储器与寄存器卷之间批量传送数据。存储器的地址由工作寄存器对指定。源地址的内容载入目标地址，然后存储器地址自动加 1，源操作数的内容不变。

LDCI 对应程序存储器，而 LDEI 对应外部数据存储器。对于程序存储器，编译器将“lrr”或“rr”的值编译成偶地址，对于数据存储器则编译成奇地址。

标志位: 没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	dst   src	2	10	E3	r	lrr

编程实例:

假如: R6 = 10H, R7 = 33H, R8 = 12H, 程序存储器地址 1033H = 0CDH, 1034H = 0C5H; 外部数据存储器地址 1033H = 0DDH, 1034H = 0D5H:

```

LDCI R8,@RR6 ; 0CDH (程序存储器地址 1033H 的内容) 载入R8
; RR6 加1 (RR6 ← RR6 + 1)
; R8 = 0CDH, R6 = 10H, R7 = 34H
LDEI R8,@RR6 ; 0DDH (数据存储器地址 1033H 的内容) 载入R8
; RR6 加1 (RR6 ← RR6 + 1)
; R8 = 0DDH, R6 = 10H, R7 = 34H

```

## 6.1.8.40 LDCPD/LDEPD — 传送数据前地址减 1 (Load Memory with Pre-Decrement)

LDCPD/LDEPD dst, src

操作:           rr ← rr - 1  
                 dst ← src

该指令用于用户栈或在程序/数据存储器 and 寄存器卷之间批量传送数据。  
存储器的地址由工作寄存器对指定并首先减 1。  
之后源地址的内容送入目标地址，源操作数的内容不变。

LDCPD 对应程序存储器，LDEPD 对应外部数据存储器。对于程序存储器，编译器将“lrr”或“rr”的值编译成偶地址，而对于数据存储器则编译成奇地址。

标志位:           没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	src   dst	2	14	F2	lrr	r

编程实例:

```

假如:           R0 = 77H, R6 = 30H, R7 = 00H:

LDCPD @RR6, R0       ;       (RR6 ← RR6 - 1)
                      ;       77H (R0 的内容) 载入程序存储器地址 2FFFH (3000H - 1H)
                      ;       R0 = 77H, R6 = 2FH, R7 = 0FFH
LDEPD @RR6, R0       ;       (RR6 ← RR6 - 1)
                      ;       77H (R0 的内容) 载入外部数据存储器地址 2FFFH (3000H - 1H)
                      ;       R0 = 77H, R6 = 2FH, R7 = 0FFH

```



## 6.1.8.41 LDCPI/LDEPI — 传送数据前地址加 1 (Load Memory with Pre-Increment)

LDCPI/LDEPI dst, src

操作:  $rr \leftarrow rr + 1$   
 $dst \leftarrow src$

该指令用于用户栈或在程序/数据存储器与寄存器卷之间批量传送数据。  
 存储器的地址由工作寄存器对指定并首先加 1。  
 之后源地址的内容载入目标地址，源操作数的内容不变。

LDCPI 对应程序存储器，LDEPI 对应外部数据存储器。对于程序存储器，编译器将“lrr”或“rr”的值编译成偶地址，而对于数据存储器则编译成奇地址。

标志位: 没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	src   dst	2	14	F3	lrr	r

编程实例:

```

假如:      R0 = 7FH, R6 = 21H, R7 = 0FFH:

LDCPI @RR6, R0      ;      (RR6 ← RR6 + 1)
                  ;      7FH (R0 的内容) 载入程序存储器地址 2200H (21FFH + 1H)
                  ;      R0 = 7FH, R6 = 22H, R7 = 00H
LDEPI @RR6, R0      ;      (RR6 ← RR6 + 1)
                  ;      7FH (R0 的内容) 载入外部数据存储器地址 2200H (21FFH + 1H)
                  ;      R0 = 7FH, R6 = 22H, R7 = 00H

```

## 6.1.8.42 LDW — 传送字数据 (Load Word)

LDW           dst, src

操作:           dst ← src

源操作数的内容（字）载入目标操作数。源操作数内容不变。

标志位:        没有标志位受影响

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	src	dst	3	8	C4	RR	RR
				8	C5	RR	IR
opc	dst	src	4	8	C6	RR	IML

编程实例:

假如:       R4 = 06H, R5 = 1CH, R6 = 05H, R7 = 02H, 寄存器 00H = 1AH,  
寄存器 01H = 02H, 寄存器 02H = 03H, 寄存器 03H = 0FH:

```
LDW  RR6,RR4    →   R6 = 06H, R7 = 1CH, R4 = 06H, R5 = 1CH
LDW  00H,02H    →   寄存器 00H = 03H, 寄存器 01H = 0FH,
                   寄存器 02H = 03H, 寄存器 03H = 0FH

LDW  RR2,@R7    →   R2 = 03H, R3 = 0FH,
LDW  04H,@01H   →   寄存器 04H = 03H, 寄存器 05H = 0FH

LDW  RR6,#1234H →   R6 = 12H, R7 = 34H
LDW  02H,#0FEDH →   寄存器 02H = 0FH, 寄存器 03H = 0EDH
```

第二个例子中, 请注意指令“LDW 00H, 02H”将源寄存器 02H 和 03H 中的内容载入目标寄存器00H和01H, 使通用寄存器 00H 中的值变为 03H, 01H 中的值变为 0FH。

其它例子演示了如何通过不同的寻址模式和格式来使用 LDW 指令。

## 6.1.8.43 MULT — 无符号数乘法 (Unsigned Multiply)

MULT dst, src

操作:  $dst \leftarrow dst \times src$ 

8 位目标操作数（寄存器对中的偶地址寄存器）与源操作数（8位）相乘，乘积（16位）保存在目标地址指定的寄存器对中。两个操作数都为无符号整型数据。

标志位:

- C:** 如果结果 >255 则置 1；否则清零
- Z:** 如果结果为“0”则置 1；否则清零
- S:** 如果结果的最高为“1”则置1；否则清零
- V:** 清零
- D:** 不受影响
- H:** 不受影响

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式	
						dst	src
opc	src	dst	3	22	84	RR	R
				22	85	RR	IR
				22	86	RR	IM

编程实例:

假如: 寄存器 00H = 20H, 寄存器 01H = 03H, 寄存器02H = 09H, 寄存器 03H = 06H:

```
MULT 00H, 02H  →  寄存器 00H = 01H, 寄存器 01H = 20H, 寄存器 02H = 09H
MULT 00H, @01H →  寄存器 00H = 00H, 寄存器 01H = 0C0H
MULT 00H, #30H →  寄存器 00H = 06H, 寄存器 01H = 00H
```

第一个例子中，指令“MULT 00H, 02H”将 8 位目标操作数（寄存器对 00H, 01H 中的 00H）与源寄存器 02H 中的操作数（09H）相乘。16 位的乘积，0120H，保存在寄存器对 00H, 01H中。

## 6.1.8.44 NEXT — Next 指令

## NEXT

操作:           PC ← @ IP  
                   IP ← IP + 2

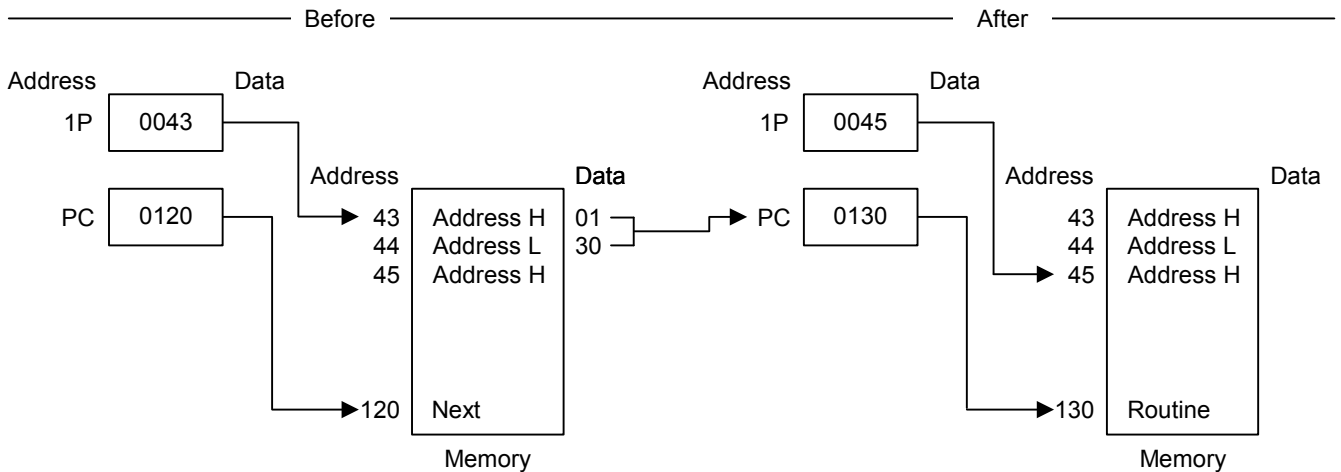
NEXT 指令在执行线性代码语言时非常有用。程序存储器中指令指针所指向的字送入 PC，并且指令指针增加 2。

标志位:        没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	10	0F

编程实例:        下图是一个如何使用 NEXT 指令的例子。



### 6.1.8.45 NOP — 空操作 (No Operation)

#### NOP

**操作:** CPU 执行这条指令时，不做任何操作。通常用顺序执行一个或多个 NOP 指令来实现一定时长的延时。

**标志位:** 没有标志位受影响

**格式:**

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	FF

**编程实例:** 在程序中执行 NOP 指令时，没有任何操作发生，只是一个指令执行时间的延时。

## 6.1.8.46 OR — 逻辑或 (Logical OR)

OR           dst, src

操作:       dst ← dst OR src

源操作数与目标操作数进行逻辑或，结果存放在目标操作数。源操作数的值不受影响。只要两个操作数中任一个的相应位为“1”，那么或的结果就是“1”，否则为“0”。

标志位:    **C:**     不受影响  
             **Z:**     如果结果是“0”则置 1；否则清零  
             **S:**     如果结果的第 7 位 (bit 7) 为“1”则置 1；否则清零  
             **V:**     总是清零  
             **D:**     不受影响  
             **H:**     不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式	
					dst	src
opc	dst   src	2	4	42	r	r
			6	43	r	lr
opc	src   dst	3	6	44	R	R
			6	45	R	IR
opc	dst   src	3	6	46	R	IM

编程实例:

假如:       R0 = 15H, R1 = 2AH, R2 = 01H, 寄存器 00H = 08H, 寄存器 01H = 37H,  
 寄存器 08H = 8AH:

```
OR   R0,R1      →   R0 = 3FH, R1 = 2AH
OR   R0,@R2     →   R0 = 37H, R2 = 01H, 寄存器 01H = 37H
OR   00H,01H    →   寄存器 00H = 3FH, 寄存器 01H = 37H
OR   01H,@00H   →   寄存器 00H = 08H, 寄存器 01H = 0BFH
OR   00H,#02H   →   寄存器 00H = 0AH
```

第一个例子中，如果工作寄存器 R0 的值是 15H，寄存器 R1 的值是 2AH，指令“OR R0, R1”将寄存器 R0 和 R1 的内容进行逻辑或，并将结果 (3FH) 存在目标寄存器 R0。

其他的例子演示了如何通过不同的寻址模式和格式来使用逻辑或指令。

## 6.1.8.47 POP — 出栈 (Pop from Stack)

POP dst

操作:           dst ← @SP  
                   SP ← SP + 1

堆栈指针指定地址的内容被装入目标操作数，然后堆栈指针加1。

标志位:         没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	8	50	R
			8	51	IR

编程实例:

假如:           寄存器 00H = 01H, 寄存器 01H = 1BH, SPH (0D8H) = 00H, SPL (0D9H) = 0FBH, 堆栈寄存器 0FBH = 55H:

POP 00H → 寄存器 00H = 55H, SP = 00FCH  
 POP @00H → 寄存器 00H = 01H, 寄存器 01H = 55H, SP = 00FCH

第一个例子中，通用寄存器 00H 的值为 01H。指令“POP 00H”将地址 00FBH 中的值（55H）送入目标寄存器 00H，然后堆栈指针加 1。寄存器 00H 的值变为 55H，并且 SP 指向地址 00FCH。





## 6.1.8.49 POPUI — 弹出用户栈 (自增) (Pop User Stack (Incrementing))

POPUI dst, src

操作: dst ← src  
IR ← IR + 1

该指令用于寄存器卷中的用户自定义栈。用户栈指针指定地址的内容载入目标寄存器，然后用户栈指针加1。

标志位: 没有标志位受影响

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	src	dst	3	8	93	R	IR

编程实例:

假如: 寄存器 00H = 01H, 寄存器 01H = 70H:

POPUI 02H, @00H → 寄存器 00H = 02H, 寄存器 01H = 70H,  
寄存器 02H = 70H

如果通用寄存器 00H 的值为 01H, 寄存器 01H 的值为 70H, 那么语句“POPUI 02H, @00H”将值70H载入目标通用寄存器 02H, 然后用户栈指针 (寄存器00H) 加1, 从 01H 变为 02H。

## 6.1.8.50 PUSH — 压栈 (Push to Stack)

**PUSH**            src

**操作:**            SP ← SP - 1  
                   @SP ← src

PUSH 指令先将栈指针减 1，然后再将源操作数的内容载入减 1 后的栈地址。此操作将在栈顶置入新的数值。

**标志位:**        没有标志位受影响

**格式:**

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	src	2	8 (内部时钟) 8 (外部时钟)	70	R
			8 (内部时钟) 8 (外部时钟)	71	IR

**编程实例:**

假如:            寄存器 40H = 4FH, 寄存器 4FH = 0AAH, SPH = 00H, SPL = 00H:

PUSH 40H    →    寄存器 40H = 4FH, 堆栈寄存器 0FFH = 4FH,  
                   SPH = 0FFH, SPL = 0FFH

PUSH @40H →    寄存器 40H = 4FH, 寄存器 4FH = 0AAH, 堆栈寄存器  
                   0FFH = 0AAH, SPH = 0FFH, SPL = 0FFH

第一个例子中，如果栈指针包含的内容为 0000H，通用寄存器 40H 的内容为 4FH，语句“PUSH 40H”使栈指针从 0000H 减为 0FFFFH，然后将寄存器 40H 的值载入地址 0FFFFH 中，即将这个新的数值加入栈顶。

## 6.1.8.51 PUSHUD — 压入用户栈 (自减) Push User Stack (Decrementing)

**PUSHUD** dst, src

**操作:** IR ← IR - 1  
dst ← src

该指令用于寄存器卷中的用户自定义栈。PUSHUD 先将用户栈指针减 1，然后将源操作数的内容载入减1后的栈指针指向的寄存器。

**标志位:** 没有标志位受影响

**格式:**

			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	dst	src	3	8	82	IR	R

**编程实例:**

假如: 寄存器 00H = 03H, 寄存器 01H = 05H, 寄存器 02H = 1AH:

PUSHUD @00H, 01H → 寄存器 00H = 02H, 寄存器 01H = 05H,  
寄存器 02H = 05H

如果用户栈指针 (例如寄存器 00H) 的值为 03H, 语句 “PUSHUD @00H, 01H” 使用户栈指针减1, 变为 02H。而寄存器 01H 的值, 05H, 则载入减 1 后的用户栈指针所指向的寄存器。



### 6.1.8.53 RCF — C 清零 (Reset Carry Flag)

**RCF**            RCF

**操作:**             $C \leftarrow 0$

进/借位标志位被无条件清零。

**标志位:**        **C:**        清除为“0”。

其他标志位都不受影响

**格式:**

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	CF

**编程实例:**

假如:             $C = \text{“1”}$  或  $\text{“0”}$  :

指令 RCF 将进/借位标志位 (C) 清零。

## 6.1.8.54 RET — 子程序返回 (Return)

## RET

操作:           PC ← @SP  
                   SP ← SP + 2

RET 指令通常被用来从 CALL 指令所调用的子程序中返回。  
 栈指针指向的地址内容被弹出到程序计数器中，  
 下一条要执行的指令地址由程序计数器的新值指定。

标志位:        没有标志位被影响

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	8 (内部堆栈) 10 (外部堆栈)	AF

编程实例:

假如:           SP = 00FCH, (SP) = 101AH, PC = 1234:

RET →        PC = 101AH, SP = 00FEH

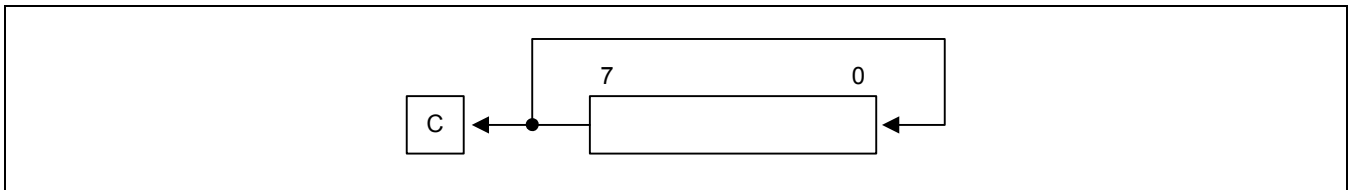
指令“RET”将堆栈指针 00FCH 中的内容 (10H) 弹出到程序计数器的高字节中，00FDH 中的内容 (1AH) 弹出到 PC 的低字节，地址 101AH 中的指令被执行。堆栈指针现在指向地址 00FEH。

## 6.1.8.55 RL — 左移 (Rotate Left)

RL            dst

操作:             $C \leftarrow \text{dst}(7)$   
                    $\text{dst}(0) \leftarrow \text{dst}(7)$   
                    $\text{dst}(n + 1) \leftarrow \text{dst}(n), n = 0-6$

目标操作数的内容左移一位，原来的第 7 位 (bit 7) 移到第 0 位 (LSB) 和 C 标志位中，如下图示：



标志位:            **C:**    如果从最高位 (bit 7) 移出的数为 “1”，则置 1  
                   **Z:**    如果结果为 “0” 则置 1；否则清零  
                   **S:**    如果结果的第 7 位 (bit 7) 为 “1” 则置 1；否则清零  
                   **V:**    如果发生溢出则置 1；否则清零  
                   **D:**    不受影响  
                   **H:**    不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	90	R
			4	91	IR

编程实例:

假如:            寄存器 00H = 0AAH, 寄存器 01H = 02H, 寄存器 02H = 17H:

RL    00H    →    寄存器 00H = 55H, C = “1”  
 RL    @01H →    寄存器 01H = 02H, 寄存器 02H = 2EH, C = “0”

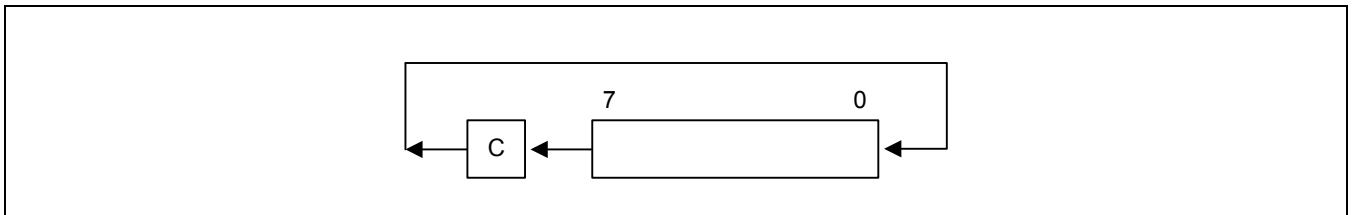
第一个例子中，如果通用寄存器 00H 的值为 0AAH (10101010B)，语句 “RL 00H” 将 0AAH 左移一位，变为 55H (01010101B)，并将进位和溢出标志置 1。

## 6.1.8.56 RLC — 带进位左移 (Rotate Left Through Carry)

RLC            dst

操作:            dst (0) ← C  
                   C ← dst (7)  
                   dst (n + 1) ← dst (n), n = 0-6

目标操作数的内容通过 C 标志位左移一位，原来的第 7 位 (bit 7) 移到 C 标志位中，原来的 C 标志位则移至第 0 位 (LSB)。



标志位:            **C:**    如果从最高位 (bit 7) 移出的为 “1” 则置 1  
                   **Z:**    如果结果为 “0” 则置 1; 否则清零  
                   **S:**    如果结果的第 7 位 (bit 7) 为 “1” 则置 1; 否则清零  
                   **V:**    如果发生溢出，也就是目标操作数的符号在移位中发生改变，则置 1; 否则清零  
                   **D:**    不受影响  
                   **H:**    不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	10	R
			4	11	IR

编程实例:

假如:            寄存器 00H = 0AAH, 寄存器 01H = 02H, 寄存器 02H = 17H, C = “0” :

RLC 00H →        寄存器 00H = 54H, C = “1”

RLC @01H →      寄存器 01H = 02H, 寄存器 02H = 2EH, C = “0”

第一个例子中，如果通用寄存器的值为 0AAH (10101010B)，语句 “RLC 00H” 将 0AAH 左移一位。原来的第 7 位 (bit 7) 将进位标志置 1，C 标志位原来的值则移至寄存器 00H 的第 0 位 (LSB)，使寄存器的值变为 54H (01010101B)。寄存器 00H 的最高位 MSB 重新置 C 标志位为 “1”，并且将溢出标志位置 1。

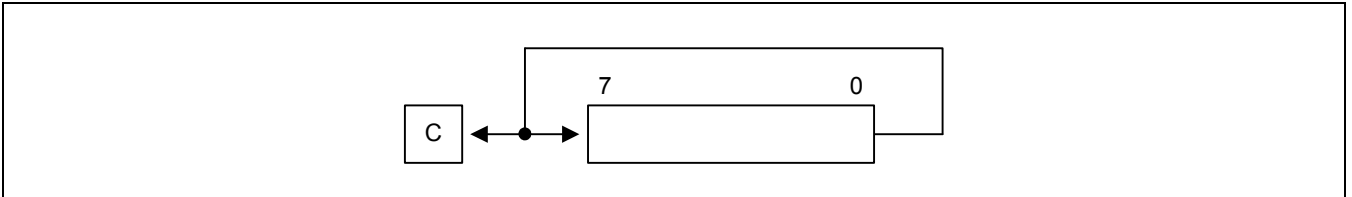


## 6.1.8.57 RR — 右移 (Rotate Right)

RR                dst

操作:            C ← dst (0)  
                   dst (7) ← dst (0)  
                   dst (n) ← dst (n + 1), n = 0-6

目标操作数的内容右移一位，原来的第 0 位 (LSB) 移到第 7 位 (MSB) 和 C 标志位中。



标志位:        **C:**    如果从第 0 位 (LSB) 移出的为“1”，则置 1  
                   **Z:**    如果结果为“0”则置 1；否则清零  
                   **S:**    如果结果的第 7 位 (bit 7) 为“1”则置 1；否则清零  
                   **V:**    如果发生溢出，也就是目标操作数的符号在移位中发生改变，则置 1；否则清零  
                   **D:**    不受影响  
                   **H:**    不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	E0	R
			4	E1	IR

编程实例:

假如:            寄存器 00H = 31H, 寄存器 01H = 02H, 寄存器 02H = 17H:

RR    00H    →    寄存器 00H = 98H, C = “1”  
 RR    @01H →    寄存器 01H = 02H, 寄存器 02H = 8BH, C = “1”

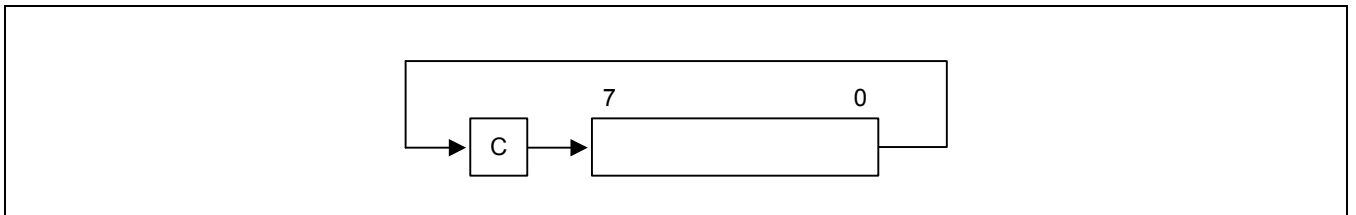
第一个例子中，如果通用寄存器 00H 的值为 31H (00110001B)，语句“RR 00H”将 00H 的值右移一位，原来第 0 位移至第 7 位，目标寄存器的值变为 98H (10011000B)。原来的第 0 位将 C 标志位置“1”，同时符号标志和溢出标志也被置“1”。

## 6.1.8.58 RRC — 带进位右移 (Rotate Right Through Carry)

RRC dst

操作: dst (7) ← C  
 C ← dst (0)  
 dst (n) ← dst (n + 1), n = 0-6

目标操作数的内容通过 C 标志位右移一位，原来的第 0 位 (LSB) 移到 C 标志位中，原来的 C 标志位则移至第 7 位 (MSB)。



标志位: **C:** 如果从最低位 (bit 0) 移出的为 “1”，则置 1  
**Z:** 如果结果为 “0” 则置 1；否则清零  
**S:** 如果结果的第 7 位 (bit 7) 为 “1” 则置 1；否则清零  
**V:** 如果发生溢出，也就是说目标操作数的符号在移位中发生改变，则置 1；否则清零  
**D:** 不受影响  
**H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	C0	R
			4	C1	IR

编程实例:

假如: 寄存器 00H = 55H, 寄存器 01H = 02H, 寄存器 02H = 17H, C = "0":

RRC 00H → 寄存器 00H = 2AH, C = "1"

RRC @01H → 寄存器 01H = 02H, 寄存器 02H = 0BH, C = "1"

第一个例子中，如果通用寄存器 00H 的值为 55H (01010101B)，语句 “RRC 00H” 将 00H 的值右移一位，原来第 0 位 (“1”) 移至进位标志位，原来的 C 标志位 (“1”) 移至第 7 位，将目标寄存器 00H 的变为 2AH (00101010B)。符号标志和溢出标志都被清零。

**6.1.8.59 SB0 — 选择 Bank 0 (Select Bank 0)****SB0**

**操作:** BANK ← 0

SB0 指令将标志位寄存器(FLAGS)中的 bank 地址标志 (FLAGS.0) 清零, 在寄存器卷的 set 1 区域选择 bank 0 。

**标志位:** 没有标志位受影响

**格式:**

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	4F

**编程实例:** 语句 SB0 将 FLAGS.0 清零, 选择 bank 0 中的寄存器进行寻址。

### 6.1.8.60 SB1 — 选择 Bank 1 (Select Bank 1)

#### SB1

**操作:** BANK ← 1

SB1 指令将标志位寄存器(FLAGS)中的 bank 地址标志 (FLAGS.0) 置 1, 在寄存器卷的 set 1 区域中选择 bank 1 。

**注释:** 某些 S3F8- 系列的单片机没有 Bank 1。

**标志位:** 没有标志位受影响

**格式:**

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	5F

**编程实例:** 指令 SB1 将 FLAGS.0 清零, 选择 bank 1 的寄存器进行寻址 (如果存在 bank 1)。

## 6.1.8.61 SBC — 带进位减法 (Subtract with Carry)

SBC dst, src

操作:  $dst \leftarrow dst - src - c$ 

目标操作数减去源操作数和当前 C 标志位的值，结果存在目标操作数中。源操作数不受影响。减法操作是通过将源操作数的补码加到目标操作数来完成的。在多次进行的精确运算中，指令允许低阶的操作数向高阶的操作数“借位”。

标志位:

- C:** 如果借位操作发生 ( $src > dst$ ) 则置 1；否则清零
- Z:** 如果结果是“0”则置 1；否则清零
- S:** 如果结果为负则置 1；否则清零
- V:** 如果发生溢出，也就是说如果操作数符号相反而差值与源操作数符号相同，则置1；否则清零
- D:** 总是被置 1
- H:** 如果从结果低4位的最高位有进位则清零；否则置 1，表示有“借位”发生。

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式		
					dst	src	
opc	dst   src	2	4	32	r	r	
			6	33	r	lr	
opc	src	3	6	34	R	R	
			6	35	R	IR	
opc	dst	src	3	6	36	R	IM

编程实例:

假如: R1 = 10H, R2 = 03H, C = “1”, 寄存器 01H = 20H, 寄存器 02H = 03H,  
寄存器 03H = 0AH:

```

SBC R1,R2      → R1 = 0CH, R2 = 03H
SBC R1,@R2    → R1 = 05H, R2 = 03H, 寄存器 03H = 0AH
SBC 01H,02H   → 寄存器 01H = 1CH, 寄存器 02H = 03H
SBC 01H,@02H  → 寄存器 01H = 15H, 寄存器 02H = 03H, 寄存器 03H = 0AH
SBC 01H,#8AH  → 寄存器 01H = 95H; C, S, V = “1”

```

第一个例子中，如果工作寄存器 R1 的值为 10H，R2 的值为 03H，语句“SBC R1, R2”从目标操作数（10H）中减去源操作数（03H）和 C 标志的值（“1”），然后将结果（0CH）存放在寄存器 R1 中。

## 6.1.8.62 SCF — C 置 1 (Set Carry Flag)

## SCF

**操作:**  $C \leftarrow 1$   
 将进位标志位 (C) 无条件置 1。

**标志位:** **C:** 置为“1”  
 其它标志位都不受影响

**格式:**

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	DF

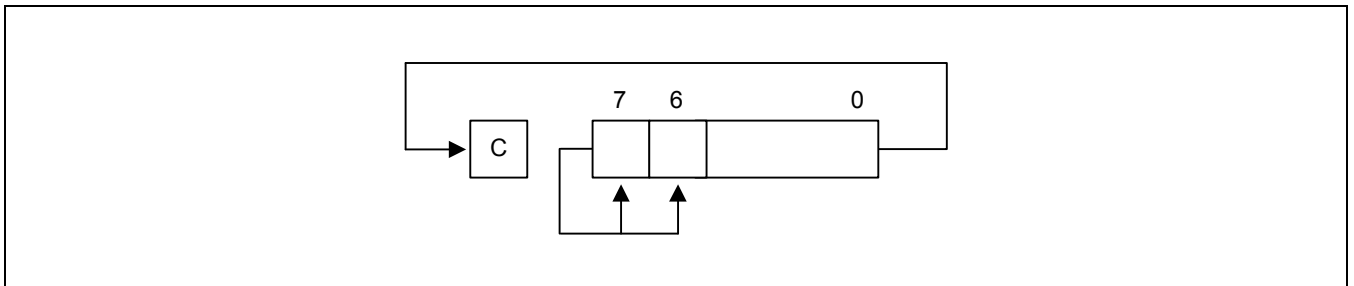
**编程实例:** 语句 SCF 将进位标志置 1。

## 6.1.8.63 SRA — 算术右移 (Shift Right Arithmetic)

SRA            dst

操作:            dst (7) ← dst (7)  
                   C        ← dst (0)  
                   dst (n) ← dst (n + 1), n = 0–6

将目标操作数算术右移一位，第 0 位 (LSB) 移至 C 标志，第 7 位 (bit 7，符号位) 改变并且移到第 6 位。



标志位:            **C:**     如果从第 0 位 (LSB) 移出的是“1”则置 1；否则清零  
                   **Z:**     如果结果是“0”则置 1；否则清零  
                   **S:**     如果结果是负数则置 1；否则清零  
                   **V:**     总是清零  
                   **D:**     不受影响  
                   **H:**     不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	D0	R
			4	D1	IR

编程实例:

假如:            寄存器 00H = 9AH, 寄存器 02H = 03H, 寄存器 03H = 0BCH, C = “1” :

SRA 00H → 寄存器 00H = 0CD, C = “0”

SRA @02H → 寄存器 02H = 03H, 寄存器 03H = 0DEH, C = “0”

第一个例子中，如果通用寄存器 00H 的值为 9AH (10011010B)，语句“SRA 00H”将寄存器00H 右移一位，第 0 位 (“0”) 清除 C 标志，第7位 (“1”) 移至第 6 位 (第 7 位保持不变)，目标寄存器 00H 的值变为 0CDH (11001101B)。

## 6.1.8.64 SRP/SRP0/SRP1 — 设置寄存器指针 (Set Register Pointer)

SRP src

SRP0 src

SRP1 src

操作:

如果 src (1) = 1 且 src (0) = 0 那么:	RP0 (3–7)	←	src (3–7)
如果 src (1) = 0 且 src (0) = 1 那么:	RP1 (3–7)	←	src (3–7)
如果 src (1) = 0 且 src (0) = 0 那么:	RP0 (4–7),	←	src (4–7),
	RP0 (3)	←	0
	RP1 (4–7)	←	src (4–7),
	RP1 (3)	←	1

源操作数的第 1 位和第 0 位 (LSB) 决定写入 2 个寄存器指针中的 1 个还是 2 个一起写入。如果两个寄存器指针都被选择, 那么选择的寄存器指针的 3 到 7 位被写入, 然后 RP0.3 清零, RP1.3 置 1。

标志位: 没有标志位受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 src
opc	src	2	4	31	IM

编程实例:

语句

SRP #40H

将地址为 0D6H 的寄存器指针 0 (RP0) 设定在 40H, 地址为 0D7H 的寄存器指针 1 (RP1) 设定在 48H。

语句“SRP0 #50H”将 RP0 设为 50H, 而语句“SRP1 #68H”将 RP1 设为 68H。



## 6.1.8.65 STOP — Stop 操作 (Stop Operation)

## STOP

**操作:** STOP 指令同时停止 CPU 时钟和系统时钟，使单片机进入 STOP 模式。在 STOP 模式下，CPU 寄存器、外设寄存器、I/O 口的控制和数据寄存器内容都保持不变。STOP 模式可以被外部的复位操作或者外部中断唤醒。对于复位操作，RESET 引脚的低电平必须保持足够长的时间，以保证晶振稳定所需的时间间隔。在应用程序中，STOP 指令后必须跟有 3 个 NOP 指令，以保证在下条指令执行之前有足够长的时间间隔来稳定晶振。如果 STOP 后没有使用 3 个或者多个 NOP 指令，内部总线的悬浮状态将会产生漏电流。

**标志位:** 没有标志位受影响

**格式:**

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	1	4	7F	-	-

**编程实例:**

```

语句
STOP          ; 停止所有单片机操作
NOP
NOP
NOP

```

**注释:** 在执行 STOP 指令之前，必须设置 STPCON 寄存器值为“10100101b”，否则 STOP 指令不会执行。

## 6.1.8.66 SUB — 减法 (Subtract)

SUB dst, src

操作:  $dst \leftarrow dst - src$ 

目标操作数减去源操作数，结果存放在目标操作数中。源操作数的内容不受影响。减法操作是将源操作数的补码加到目标操作数来完成的。

标志位:

- C:** 如果“借位”发生则置 1；否则清零
- Z:** 如果结果是“0”则置 1；否则清零
- S:** 如果结果是负数则置 1；否则清零
- V:** 如果发生溢出，也就是如果操作数符号相反而结果与源操作数的符号相同，则置 1；否则清零
- D:** 总是被置 1
- H:** 如果从结果低 4 位的最高位有进位则清零；否则置 1，表示有“借位”发生。

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式	
					dst	src
opc	dst   src	2	4	22	r	r
			6	23	r	lr
opc	src	3	6	24	R	R
			6	25	R	IR
opc	dst	3	6	26	R	IM

编程实例:

假如: R1 = 12H, R2 = 03H, 寄存器 01H = 21H, 寄存器 02H = 03H, 寄存器 03H = 0AH:

```

SUB R1,R2      → R1 = 0FH, R2 = 03H
SUB R1,@R2    → R1 = 08H, R2 = 03H
SUB 01H,02H   → 寄存器 01H = 1EH, 寄存器 02H = 03H
SUB 01H,@02H  → 寄存器 01H = 17H, 寄存器 02H = 03H
SUB 01H,#90H  → 寄存器 01H = 91H; C, S, V = "1"
SUB 01H,#65H  → 寄存器 01H = 0BCH; C 和 S = "1", V = "0"

```

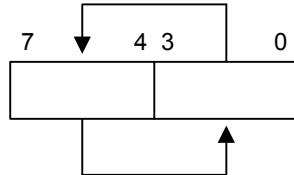
在第一个例子中，如果工作寄存器 R1 的值为 12H，R2 为 03H，语句“SUB R1, R2”将源操作数（03H）从目标操作数（12H）中减去，并将结果（0FH）存在目标寄存器 R1 中。

## 6.1.8.67 SWAP — 交换 (Swap Nibbles)

SWAP dst

操作: dst (0 - 3) ↔ dst (4 - 7)

目标操作数的低四位和高四位互相交换。



标志位:

- C:** 没有定义
- Z:** 如果结果是“0”则置 1；否则清零
- S:** 如果结果的第 7 位 (bit 7) 为“1”则置 1；否则清零
- V:** 没有定义
- D:** 不受影响
- H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc	dst	2	4	F0	R
			4	F1	IR

编程实例:

假如: 寄存器 00H = 3EH, 寄存器 02H = 03H, 寄存器 03H = 0A4H:

SWAP 00H → 寄存器 00H = 0E3H

SWAP @02H → 寄存器 02H = 03H, 寄存器 03H = 4AH

第一个例子中, 如果通用寄存器 00H 的值为 3EH (00111110B), 语句“SWAP 00H”将其高四位和低四位交换, 使寄存器 00H 的值变为 0E3H (11100011B)。

## 6.1.8.68 TCM — 取反后位测试 (Test Complement Under Mask)

TCM dst, src

操作: (NOT dst) AND src

该指令用于测试目标操作数中特定位是否为逻辑1。

被测试的位通过将源操作数中相应位设为“1”屏蔽来指定。TCM 语句将目标操作数取反，然后与源操作数进行与操作。通过检查零标志位 (Z) 来确定结果。

目标操作数和源操作数不受影响。

标志位:

- C:** 不受影响
- Z:** 如果结果是“0”则置 1；否则清零
- S:** 如果结果的第 7 位 (bit 7) 为“1”则置 1；否则清零
- V:** 总是清零
- D:** 不受影响
- H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src
opc	dst   src	2	4	62	r	r
			6	63	r	lr
opc	src	3	6	64	R	R
			6	65	R	IR
opc	dst	3	6	66	R	IM

编程实例:

假如: R0 = 0C7H, R1 = 02H, R2 = 12H, 寄存器 00H = 2BH, 寄存器 01H = 02H, 寄存器 02H = 23H:

```
TCM R0,R1    → R0 = 0C7H, R1 = 02H, Z = "1"
TCM R0,@R1   → R0 = 0C7H, R1 = 02H, 寄存器 02H = 23H, Z = "0"
TCM 00H,01H  → 寄存器 00H = 2BH, 寄存器 01H = 02H, Z = "1"
TCM 00H,@01H → 寄存器 00H = 2BH, 寄存器 01H = 02H,
                寄存器 02H = 23H, Z = "1"
TCM 00H,#34  → 寄存器 00H = 2BH, Z = "0"
```

第一个例子中，如果工作寄存器 R0 的值为 0C7H (11000111B)，寄存器 R1 为 02H (00000010B)，语句“TCM R0, R1”测试目标寄存器的第 1 位 (bit 1) 是否为“1”。由于屏蔽值与对应的测试位一致，Z 标志被置 1，可用来确定 TCM 操作的结果。

## 6.1.8.69 TM — 位测试 (Test Under Mask)

TM dst, src

操作: dst AND src

该指令用于测试目标操作数中特定位是否为逻辑 0。  
 被测试的位通过将源操作数中相应位设为“1”（屏蔽）来指定，  
 然后目标操作数与源操作数进行与操作。通过检查零标志位（Z）来确定结果。  
 目标操作数和源操作数不受影响。

标志位: **C:** 不受影响  
**Z:** 如果结果是“0”则置 1；否则清零  
**S:** 如果结果的第 7 位为“1”则置 1；否则清零  
**V:** 总是清零  
**D:** 不受影响  
**H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	src	
opc	dst   src	2	4	72	r	r	
			6	73	r	lr	
opc	src	3	6	74	R	R	
			6	75	R	IR	
opc	dst	src	3	6	76	R	IM

编程实例:

假如: R0 = 0C7H, R1 = 02H, R2 = 18H, 寄存器 00H = 2BH, 寄存器 01H = 02H,  
 寄存器 02H = 23H:

```

TM R0,R1    → R0 = 0C7H, R1 = 02H, Z = "0"
TM R0,@R1   → R0 = 0C7H, R1 = 02H, 寄存器 02H = 23H, Z = "0"
TM 00H,01H  → 寄存器 00H = 2BH, 寄存器 01H = 02H, Z = "0"
TM 00H,@01H → 寄存器 00H = 2BH, 寄存器 01H = 02H,
               寄存器 02H = 23H, Z = "0"
TM 00H,#54H → 寄存器 00H = 2BH, Z = "1"
  
```

第一个例子中，如果工作寄存器 R0 的值为 0C7H (11000111B)，寄存器 R1 的值为 02H (00000010B)，语句“TM R0, R1”测试目标寄存器的第1位是否为“0”。  
 由于屏蔽值与对应的的测试位不匹配，Z 标志被清零，可以用来确定 TM 操作的结果。

6.1.8.70 WFI — 等待中断 (Wait for Interrupt)

WFI

**操作:** 在等待状态时，CPU 被暂停直到中断发生，只有 DMA 传送仍然可以工作。  
WFI 状态可以由内部中断，包括快速中断唤醒。

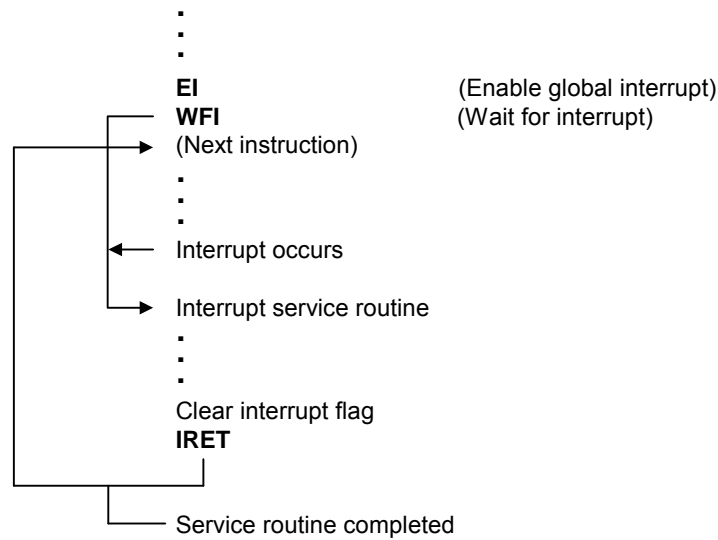
**标志位:** 没有标志位受影响

**格式:**

	字节数	时钟周期	指令代码 (Hex)
<div style="border: 1px solid black; display: inline-block; padding: 2px 10px;">opc</div>	1	4n	3F

(n = 1, 2, 3, ...)

**编程实例:** 以下实例程序的结构表示了“WFI”指令执行后的一系列操作：



## 6.1.8.71 XOR — 逻辑异或 (Logical Exclusive OR)

XOR dst, src

操作: dst ← dst XOR src

源操作数与目标操作数进行逻辑异或，结果存在目标操作数中。  
如果两个操作数中对应的位不相同，那么异或的结果为“1”；否则为“0”。

标志位: **C:** 不受影响  
**Z:** 如果结果是“0”则置 1；否则清零  
**S:** 如果结果的第 7 位 (bit 7) 为“1”则置 1；否则清零  
**V:** 总是清零  
**D:** 不受影响  
**H:** 不受影响

格式:

		字节数	时钟周期	指令代码 (Hex)	寻址模式	
					dst	src
opc	dst   src	2	4	B2	r	r
			6	B3	r	lr
opc	src	3	6	B4	R	R
			6	B5	R	IR
opc	dst	3	6	B6	R	IM

编程实例:

假如: R0 = 0C7H, R1 = 02H, R2 = 18H, 寄存器 00H = 2BH, 寄存器 01H = 02H,  
寄存器 02H = 23H:

```

XOR R0,R1      → R0 = 0C5H, R1 = 02H
XOR R0,@R1    → R0 = 0E4H, R1 = 02H, 寄存器 02H = 23H
XOR 00H,01H   → 寄存器 00H = 29H, 寄存器 01H = 02H
XOR 00H,@01H  → 寄存器 00H = 08H, 寄存器 01H = 02H, 寄存器 02H = 23H
XOR 00H,#54H  → 寄存器 00H = 7FH

```

第一个例子中，如果工作寄存器 R0 的值为 0C7H，R1 的值为 02H，语句“XOR R0, R2”将R1和R0的值进行逻辑异或，并将结果（0C5H）存在目标寄存器 R0 中。

# 7 时钟电路

## 7.1 概述

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 的外部晶振的频率范围为0.4MHz 到11.1MHz，最大CPU时钟频率为11.1 MHz。通过X<sub>IN</sub> 和 X<sub>OUT</sub> 管脚将振荡器或外部时钟源接入到芯片内。

### 7.1.1 系统时钟电路

系统时钟电路由下列部分组成：

- 外部晶振，陶瓷谐振器，RC 振荡源或外部时钟源
- 振荡器停止和唤醒功能
- CPU 时钟可通过编程进行分频 (f<sub>xx</sub> 1, 2, 8, 16分频)
- 系统时钟控制寄存器 CLKCON
- 时钟控制寄存器 OSCCON 和 STOP 控制寄存器 STPCON

### 7.1.2 CPU 时钟助记符

本文中，下列助记符用来描述 CPU 时钟：

- fx: 主时钟
- fxt: 副时钟
- fxx: 选定的系统时钟



7.1.3 主振荡器电路

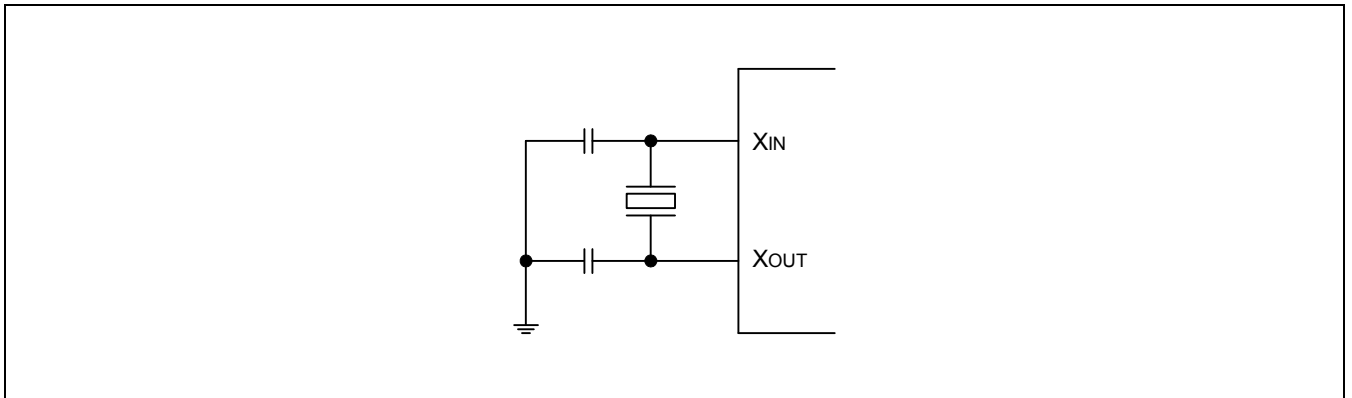


图 7-1 晶体/陶瓷振荡器 (fX)

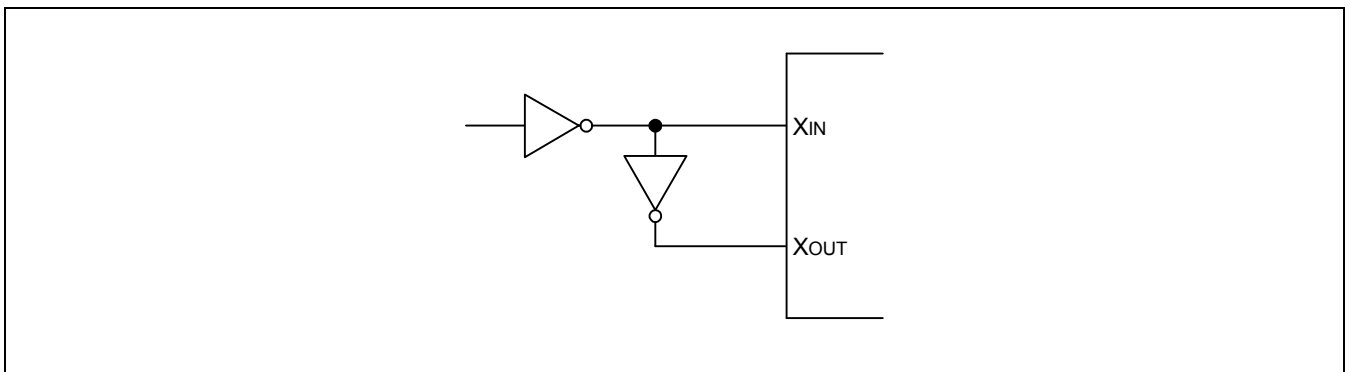


图 7-2 外部振荡器 (fX)

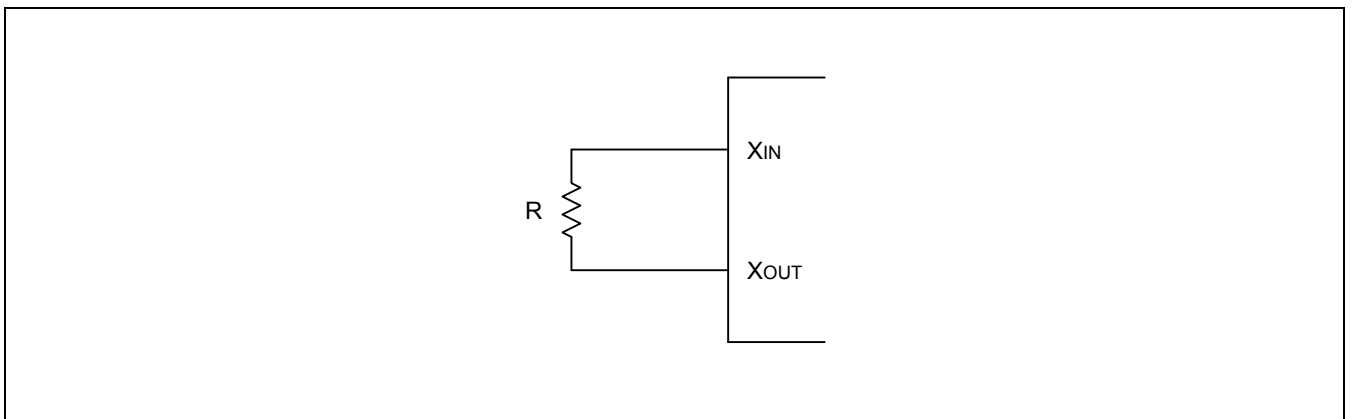


图 7-3 RC 振荡器 (fX)

7.1.4 副振荡器电路

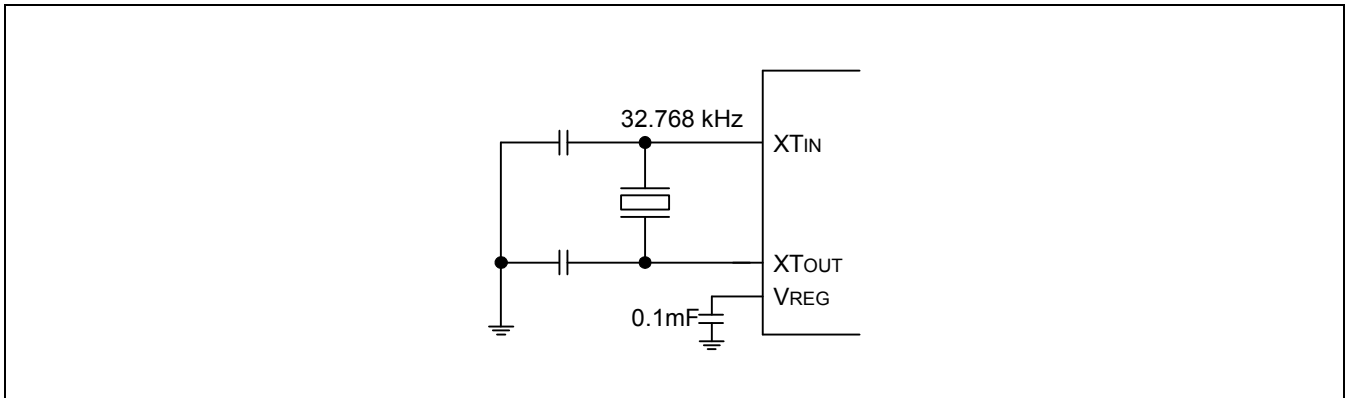


图 7-4 晶体振荡器 (fxt)

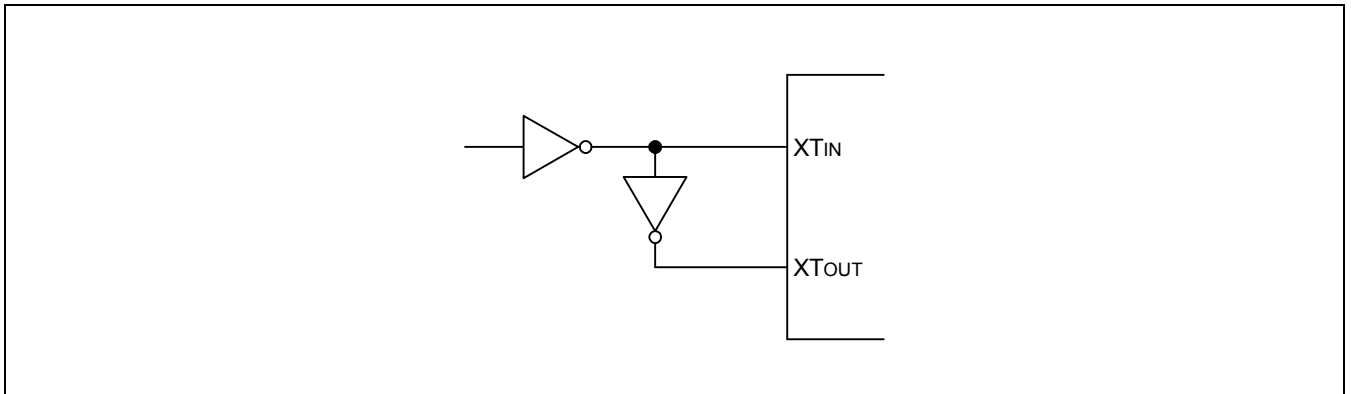


图 7-5 外部振荡器 (fxt)

### 7.1.5 省电模式下时钟状态

两种省电模式，STOP 模式和 IDLE 模式，分别对时钟振荡产生如下影响：

- 在STOP模式下，主振荡器停止工作。在复位操作或具有RC延时噪声滤波器的外部中断被触发下，CPU会退出STOP模式，振荡器重新起振。副振荡器工作且watch timer的时钟源是副时钟时，CPU也能退出STOP模式。
- 在IDLE模式下，内部时钟信号进入CPU的通路被切断，但中断及Timer仍然工作。复位或中断(外部中断或内部中断)都可以使CPU退出IDLE模式。

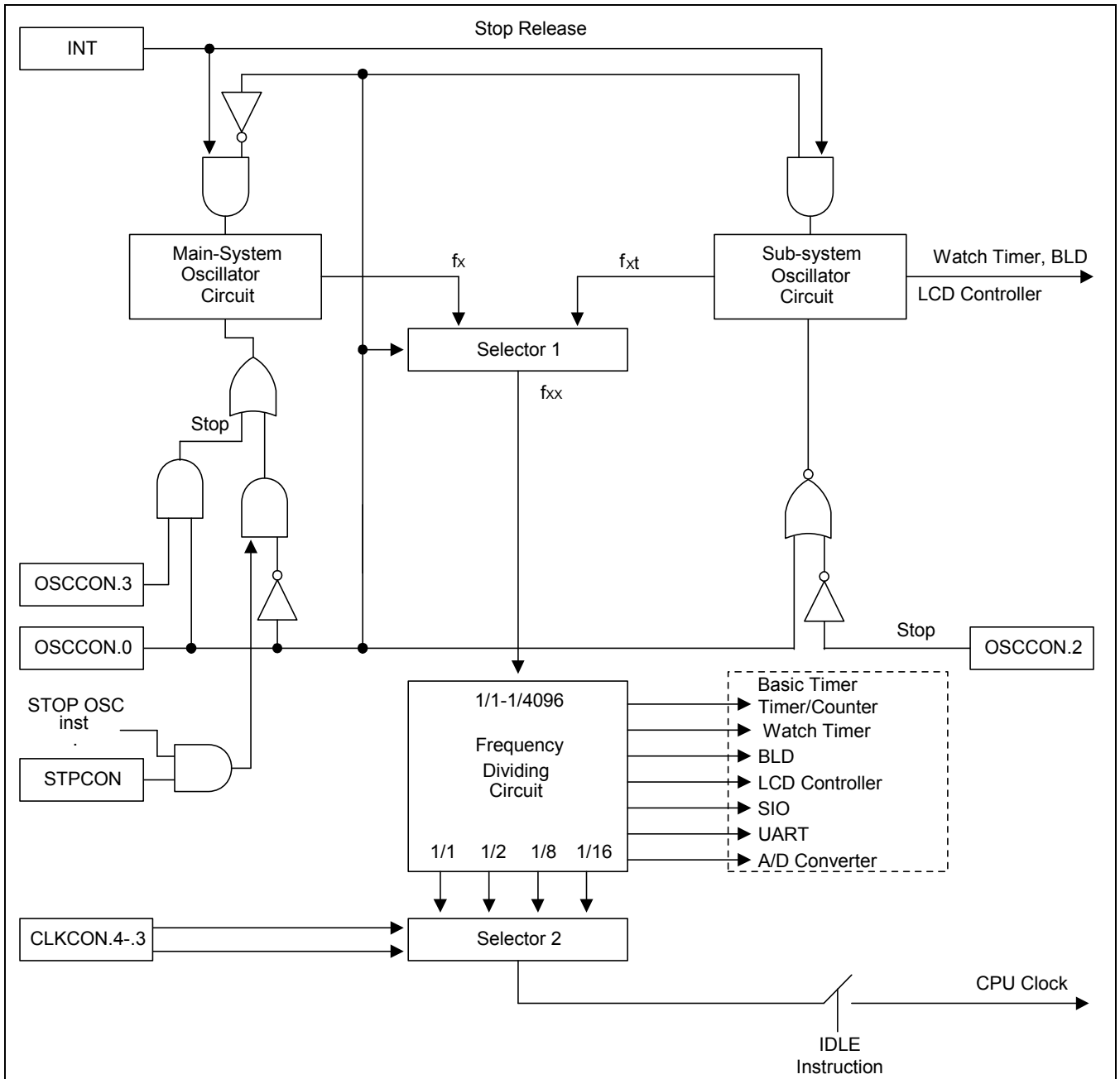


图 7-6 系统时钟电路框图

### 7.1.6 系统时钟控制寄存器 (CLKCON)

系统时钟控制寄存器CLKCON的地址是set 1的D4H，可读/写，有如下功能：

- 系统时钟分频系数选择

主振荡器建立以后， $f_{OSC}/16$ （最慢时钟速度）被选择为CPU时钟。如有需要，也可提高CPU时钟速度至 $f_{OSC}$ ， $f_{OSC}/2$  或  $f_{OSC}/8$ 。

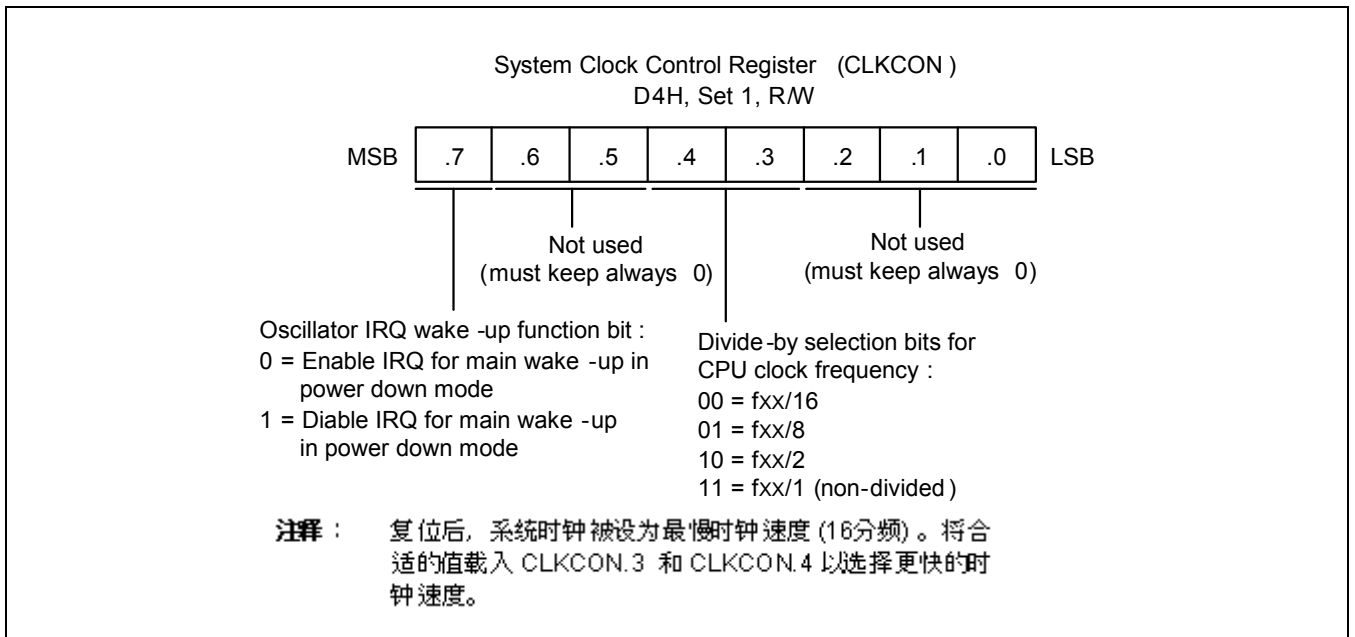


图 7-7 系统时钟控制寄存器 (CLKCON)

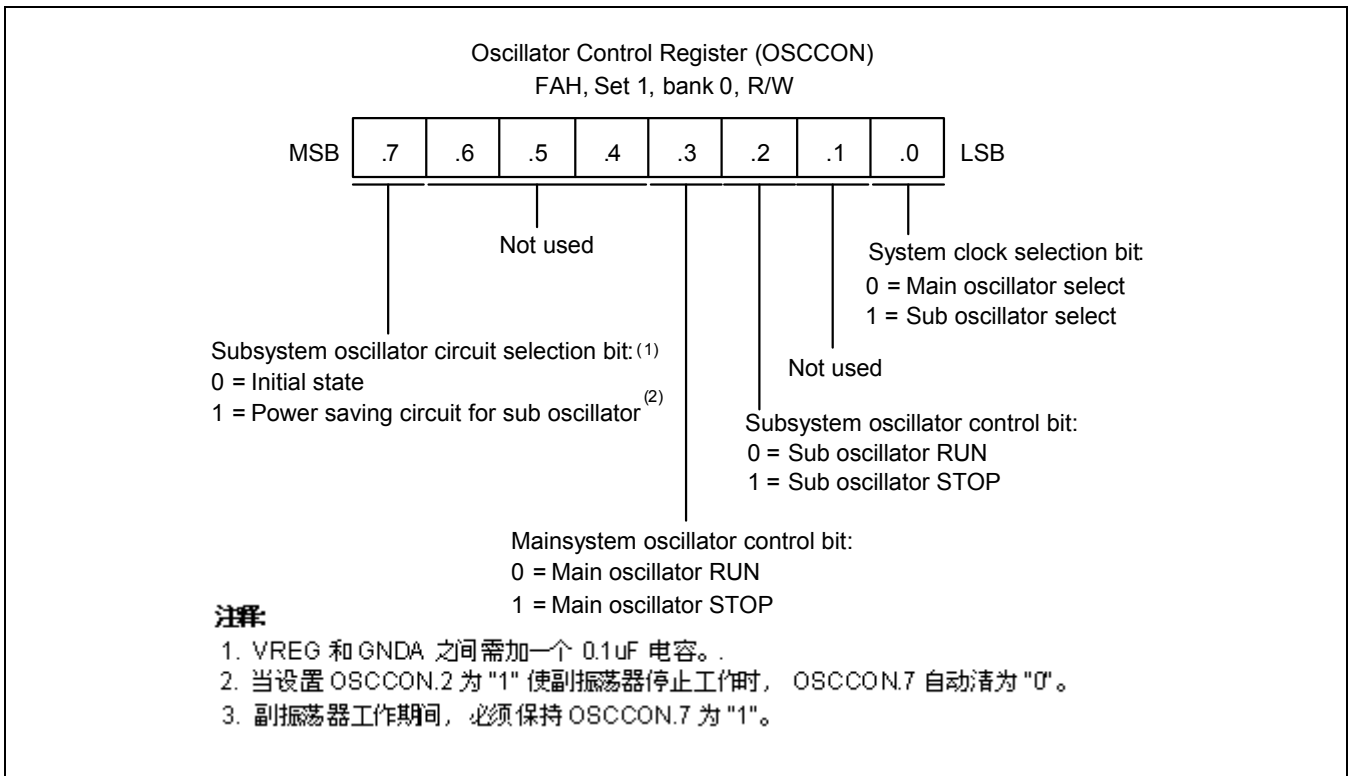


图 7-8 时钟控制寄存器 (OSCCON)

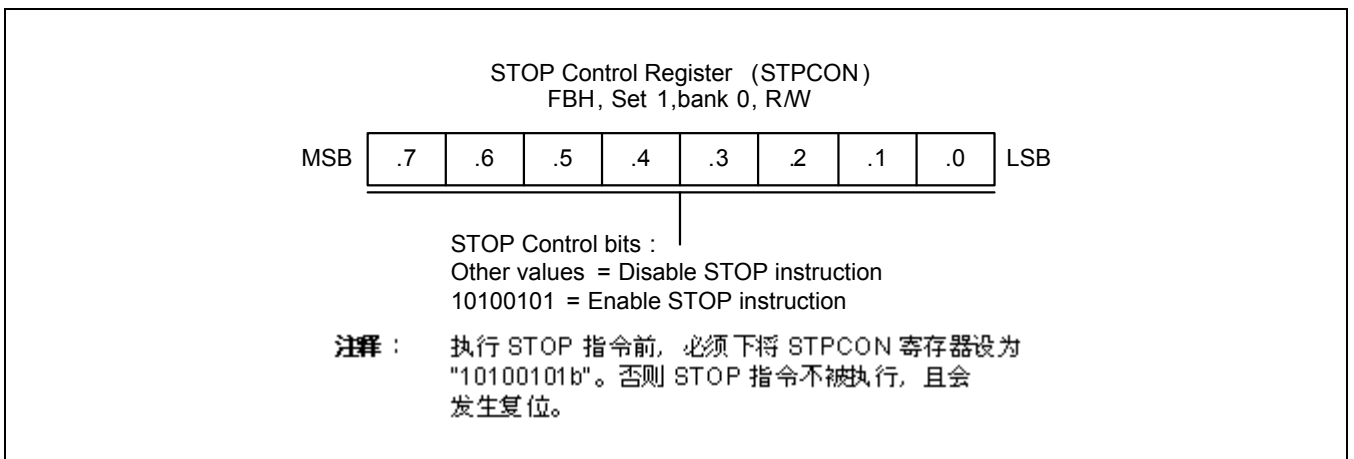


图 7-9 STOP 控制寄存器 (STPCON)

### 7.1.7 CPU 时钟切换

载入振荡器控制寄存器OSCCON

的值，决定了CPU时钟选择主时钟或副时钟；设置CLKCON寄存器，决定时钟分频。这就实现了在主副时钟之间动态切换以及工作频率的改变。

OSCCON.0 为CPU时钟选择主时钟(fx) 或副时钟(fxt)。 OSCCON .3启动或停止主时钟振荡，OSCCON.2启动或停止副时钟振荡。CLKCON.4-.3控制分频电路，将选定的fxx时钟1, 2, 8, 16分频。

例如，需要将CPU时钟从默认值fx (普通工作模式，主时钟为fx/16)， 切换到副时钟并停止主时钟。为此，可将CLKCON.4-.3设置为"11"，将OSCCON.0 设置为"1"，同时将OSCCON.3 设置为"1"。这就把时钟从fx切换到fxt，并停止了主时钟振荡。

从副时钟切换到主时钟时，必须遵循以下步骤：首先，将OSCCON.3 设为"0"以启用主时钟振荡。然后，在特定的机器周期之后，将OSCCON.0 设置为"0"来选择主时钟。

#### 编程实例 7-1 CPU时钟切换

1. 本例为如何从主时钟切换到副时钟：

```
MA2SUB          LD      OSCCON, #01H      ;      切换到副时钟
                                     ;      停止主时钟振荡
```

```
RET
```

2. 本例为如何从副时钟切换到主时钟：

```
SUB2MA         AND     OSCCON, #07H      ;      启动主时钟振荡
                CALL   DLY16            ;      延时 16 ms
                AND     OSCCON, #06H      ;      切换到主时钟
                RET
DLY16          SRP     #0C0H
                LD      R0, #20H
DEL            NOP
                DJNZ   R0, DEL
                RET
```

# 8

## 复位和省电模式

### 8.1 系统复位

#### 8.1.1 概述

上电复位时， $V_{DD}$ 电压上升到高电平而RESET管脚被强制拉到低电平。RESET信号通过一个施密特触发器电路输入，并与CPU时钟同步。这个过程将S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X带入既定的工作状态。

上电后到误差范围内的最小时间间隔里，RESET管脚必须保持低电平，来为CPU时钟振荡提供稳定时间。振荡稳定所需的最小复位操作时间是1ms。

正常工作状态下 ( $V_{DD}$ 和RESET都是高电平)，无论何时发生复位，nRESET管脚都将被强制拉到低电平并启动复位操作。所有的系统和外设控制寄存器将同时被复位成默认值。

概括起来，复位操作时如下事件依次发生：

- 禁止所有中断。
- 看门狗功能启动 (basic timer)。
- P0-8 设为输入模式，所有I/O口的上拉电阻禁止。
- 外设控制和数据寄存器设置被禁止，并复位成默认值。
- 程序计数器 (PC) 被置为ROM中的程序复位地址 0100H。
- 当设定的振荡稳定时间结束以后，正常模式下ROM地址0100H (以及 0101H)中的指令被取值和执行。
- 在S3F828B (Full-Flash产品)中，ROM中的复位地址可在Smart Option中改变。详情请参考第19章嵌入式闪存接口。

#### 8.1.2 正常模式复位操作

正常模式下，Test管脚接到 $V_{SS}$ 。复位后，S3C828BX(64-K字节)，S3C8289X (32-K字节)，和S3C8285X (16K-字节) 的片上ROM访问被使能(外部接口非自动配置)。

**注释：** 为改变振荡稳定时间，可在进入 STOP 模式前，将合适的配置写入 basic timer 控制寄存器 BTCON 中。同样，若不使用 basic timer 的看门狗功能 (当 basic timer 计数器溢出时，系统复位)，可在 BTCON 的高位写入"1010B"来禁止该功能。

### 8.1.3 硬件复位值

表 8-1, 表 8-2, 表 8-3 列出了复位操作后CPU和系统寄存器, 外设控制寄存器, 以及外设数据寄存器的复位值。用以下的助记符来代表复位值:

- "1" 或 "0" 代表复位后该位的值相应的为逻辑1或逻辑0。
- "x" 代表复位后该位的值不确定。
- 横线 ("-") 代表该位未使用或者未映射, 但读取的值为0。

表 8-1 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X Set 1 寄存器及其复位后初始值

寄存器名	助记标号	地址		复位后各位的值								
		Dec	Hex	7	6	5	4	3	2	1	0	
地址空间 D0H–D2H 未映射												
Basic Timer 控制寄存器	BTCON	211	D3H	0	0	0	0	0	0	0	0	0
系统时钟控制寄存器	CLKCON	212	D4H	0	-	-	0	0	-	-	-	-
系统标志寄存器	FLAGS	213	D5H	x	x	x	x	x	x	0	0	0
寄存器指针 0	RP0	214	D6H	1	1	0	0	0	-	-	-	-
寄存器指针 1	RP1	215	D7H	1	1	0	0	1	-	-	-	-
堆栈指针 (高字节)	SPH	216	D8H	x	x	x	x	x	x	x	x	x
堆栈指针 (低字节)	SPL	217	D9H	x	x	x	x	x	x	x	x	x
指令指针 (高字节)	IPH	218	DAH	x	x	x	x	x	x	x	x	x
指令指针 (低字节)	IPL	219	DBH	x	x	x	x	x	x	x	x	x
中断请求寄存器	IRQ	220	DCH	0	0	0	0	0	0	0	0	0
中断屏蔽寄存器	IMR	221	DDH	x	x	x	x	x	x	x	x	x
系统模式寄存器	SYM	222	DEH	0	-	-	x	x	x	0	0	0
寄存器页指针	PP	223	DFH	0	0	0	0	0	0	0	0	0

**注释:**

1. 'x' 表示该位的复位值不确定
2. '-' 表示 该位未使用或未映射, 但读取该位时, 读到的值为"0".



表 8-2 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X Set 1, Bank 0 寄存器及其复位后初始值

寄存器名	助记标记	地址		复位后各位的值								
		Dec	Hex	7	6	5	4	3	2	1	0	
LCD 控制寄存器	LCON	208	D0H	0	0	0	0	0	0	0	0	0
钟表定时器控制寄存器	WTCON	209	D1H	0	0	0	0	0	0	0	0	0
电池电压检测控制寄存器	BLDCON	210	D2H	–	–	0	0	0	0	0	0	0
SIO 控制寄存器	SIOCON	224	E0H	0	0	0	0	0	0	0	0	0
SIO 数据寄存器	SIODATA	225	E1H	0	0	0	0	0	0	0	0	0
SIO 预处理寄存器	SIOPS	226	E2H	0	0	0	0	0	0	0	0	0
Timer 0 控制寄存器	T0CON	227	E3H	0	0	0	0	0	0	0	0	0
Timer 0 计数器(高字节)	T0CNTH	228	E4H	0	0	0	0	0	0	0	0	0
Timer 0 计数器(低字节)	T0CNTL	229	E5H	0	0	0	0	0	0	0	0	0
Timer 0 数据寄存器(高字节)	T0DATAH	230	E6H	1	1	1	1	1	1	1	1	1
Timer 0 数据寄存器(低字节)	T0DATAL	231	E7H	1	1	1	1	1	1	1	1	1
Timer A 控制寄存器	TACON	232	E8H	0	0	0	0	0	0	0	0	0
Timer A 计数器	TACNT	233	E9H	0	0	0	0	0	0	0	0	0
Timer A 数据寄存器	TADATA	234	EAH	1	1	1	1	1	1	1	1	1
Timer 1 控制寄存器	T1CON	235	EBH	0	0	0	0	0	0	0	0	0
Timer 1 计数器(高字节)	T1CNTH	236	ECH	0	0	0	0	0	0	0	0	0
Timer 1 计数器(低字节)	T1CNTL	237	EDH	0	0	0	0	0	0	0	0	0
Timer 1 数据寄存器(高字节)	T1DATAH	238	EEH	1	1	1	1	1	1	1	1	1
Timer 1 数据寄存器(低字节)	T1DATAL	239	EFH	1	1	1	1	1	1	1	1	1
Timer B 数据寄存器(高字节)	TBDATAH	240	F0H	1	1	1	1	1	1	1	1	1
Timer B 数据寄存器(低字节)	TBDATAL	241	F1H	1	1	1	1	1	1	1	1	1
Timer B 控制寄存器	TBCON	242	F2H	0	0	0	0	0	0	0	0	0
A/D 转换控制寄存器	ADCON	243	F3H	–	0	0	0	0	0	0	0	0
A/D 转换数据寄存器(高字节)	ADDATAH	244	F4H	x	x	x	x	x	x	x	x	x
A/D 转换数据寄存器(低字节)	ADDATAL	245	F5H	–	–	–	–	–	–	–	x	x
UART 控制寄存器	UARTCON	246	F6H	0	0	0	0	0	0	0	0	0
UART 数据寄存器	UDATA	247	F7H	x	x	x	x	x	x	x	x	x
UART 波特率数据寄存器	BRDATA	248	F8H	1	1	1	1	1	1	1	1	1
中断标志寄存器	INTPND	249	F9H	–	–	0	0	0	0	0	0	0
时钟控制寄存器	OSCCON	250	FAH	0	–	–	–	0	0	–	0	0
STOP 控制寄存器	STPCON	251	FBH	0	0	0	0	0	0	0	0	0
地址空间 FCH 未映射												
Basic Timer 计数器	BTCNT	253	FDH	0	0	0	0	0	0	0	0	0
地址空间 FEH未映射												
中断优先级寄存器	IPR	255	FFH	x	x	x	x	x	x	x	x	x

表 8-3 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X Set 1, Bank 1 寄存器及其复位后初始值

寄存器名	助记标记	地址		复位后各位的值								
		Dec	Hex	7	6	5	4	3	2	1	0	
闪存扇区地址寄存器(高字节)	FMSECH	208	D0H	0	0	0	0	0	0	0	0	0
闪存扇区地址寄存器(低字节)	FMSECL	209	D1H	0	0	0	0	0	0	0	0	0
闪存控制寄存器	FMCON	210	D2H	0	0	0	0	0	0	–	–	0
P0 口控制寄存器 (高字节)	P0CONH	224	E0H	0	0	0	0	0	0	0	0	0
P0 口控制寄存器 (低字节)	P0CONL	225	E1H	0	0	0	0	0	0	0	0	0
P0 口中断控制寄存器 (高字节)	P0INTH	226	E2H	0	0	0	0	0	0	0	0	0
P0 口中断控制寄存器 (低字节)	P0INTL	227	E3H	0	0	0	0	0	0	0	0	0
P0 口中断标志寄存器	P0PND	228	E4H	0	0	0	0	0	0	0	0	0
P1 口控制寄存器 (高字节)	P1CONH	229	E5H	–	–	0	0	0	0	0	0	0
P1 口控制寄存器 (低字节)	P1CONL	230	E6H	0	0	0	0	0	0	0	0	0
P1 口上拉电阻使能控制寄存器	P1PUR	231	E7H	–	0	0	0	0	0	0	0	0
P2 口控制寄存器 (高字节)	P2CONH	232	E8H	0	0	0	0	0	0	0	0	0
P2 口控制寄存器 (低字节)	P2CONL	233	E9H	0	0	0	0	0	0	0	0	0
P3 口控制寄存器 (高字节)	P3CONH	234	EAH	–	–	0	0	0	0	0	0	0
P3 口控制寄存器 (低字节)	P3CONL	235	EBH	0	0	0	0	0	0	0	0	0
P4 口控制寄存器 (高字节)	P4CONH	236	ECH	0	0	0	0	0	0	0	0	0
P4 口控制寄存器 (低字节)	P4CONL	237	EDH	0	0	0	0	0	0	0	0	0
P4 口上拉电阻使能控制寄存器	P4PUR	238	EEH	0	0	0	0	0	0	0	0	0
P5 口上拉电阻使能控制寄存器	P5PUR	239	EFH	0	0	0	0	0	0	0	0	0
P0 口数据寄存器	P0	240	F0H	0	0	0	0	0	0	0	0	0
P1 口数据寄存器	P1	241	F1H	–	0	0	0	0	0	0	0	0
P2 口数据寄存器	P2	242	F2H	0	0	0	0	0	0	0	0	0
P3 口数据寄存器	P3	243	F3H	–	–	0	0	0	0	0	0	0
P4 口数据寄存器	P4	244	F4H	0	0	0	0	0	0	0	0	0
P5 口数据寄存器	P5	245	F5H	0	0	0	0	0	0	0	0	0
P6 口数据寄存器	P6	246	F6H	0	0	0	0	0	0	0	0	0
P7 口数据寄存器	P7	247	F7H	–	–	–	–	0	0	0	0	0
P8 口数据寄存器	P8	248	F8H	0	0	0	0	0	0	0	0	0
P5 口控制寄存器 (高字节)	P5CONH	249	F9H	0	0	0	0	0	0	0	0	0
P5 口控制寄存器 (低字节)	P5CONL	250	FAH	0	0	0	0	0	0	0	0	0
P6 口控制寄存器 (高字节)	P6CONH	251	FBH	0	0	0	0	0	0	0	0	0
P6 口控制寄存器 (低字节)	P6CONL	252	FCH	0	0	0	0	0	0	0	0	0
P7 口控制寄存器	P7CON	253	FDH	0	0	0	0	0	0	0	0	0
P8 口控制寄存器	P8CON	254	FEH	0	0	0	0	0	0	0	0	0
闪存用户可编程使能寄存器	FMUSR	255	FFH	0	0	0	0	0	0	0	0	0

## 8.2 省电模式

### 8.2.1 STOP 模式

STOP 指令（指令码 7FH）将会使系统进入 STOP 模式。在 STOP 模式下，CPU 和所有外设将停止工作。也就是说，片上主振荡器会停止，芯片消耗电流将减小到  $3\mu\text{A}$  以下。当时钟冻结时，所有的系统功能停止，但存储在内部寄存器卷中的数据仍然保留。可以通过 2 种方式退出 STOP 模式：复位或中断（详 [图 7-6](#)）。

**注释：** 使用外部时钟源时，不能使用 STOP 模式。因为  $X_{\text{IN}}$  或  $XT_{\text{IN}}$  管脚必须内部连接到  $V_{\text{SS}}$  以减小漏电流。

### 8.2.2 使用 NRESET 退出 STOP 模式

nRESET 信号被释放并恢复高电平时，退出 STOP 模式：

所有系统和外设控制寄存器恢复为默认值，数据寄存器仍被保持。因为 CLKCON.3 和 CLKCON.4 被清除为“00B”，CPU 时钟选择最慢频率（ $f_{\text{xx}}/16$ ）。振荡稳定时间间隔之后，CPU 调入 ROM 地址 0100H（以及 0101H）的指令开始系统初始化。

### 8.2.3 使用外部中断退出 STOP 模式

具有 RC 延迟噪声滤波电路的外部中断可以让系统退出 STOP 模式。

CPU 当前的内部工作模式决定了退出 STOP 可用的中断类型。S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 中断结构中可用作的退出 STOP 模式的外部中断为：

- 外部中断 P0.0–P0.7 (INT0–INT7)

关于 STOP 模式的退出请注意以下条件：

- 若使用外部中断退出 STOP 模式，除了 STPCON 寄存器之外的所有系统和外设控制寄存器都不发生改变。
- 若使用内部或外部中断退出 STOP 模式，也可对振荡稳定时间进行编程。但必须在进入 STOP 模式之前将控制和时钟设置成合适的值。
- 当使用外部中断退出 STOP 模式时，CLKCON.4 和 CLKCON.3 的设置不发生改变，当前选择的时钟值也将保留。
- 在退出 STOP 模式时，系统处理外部中断服务程序。中断返回后，执行紧接着 STOP 令的下一条指令。

### 8.2.4 使用内部中断退出 STOP 模式

激活任意已使能的中断，可退出 STOP 模式。其他内容和外部中断一样。

### 8.2.5 如何进入 STOP 模式

先写STPCON 寄存器然后写 STOP 指令(保持这个顺序)。

```
LD    STPCON,#10100101B
STOP
NOP
NOP
NOP
```

### 8.2.6 IDLE 模式

IDLE 指令（指令码 6FH）将会使系统进入 IDLE 模式。在 IDLE 模式下，CPU 停止但某些外设仍然工作。IDLE 模式下，内部时钟信号无法进入CPU，但所有外设的时钟仍然有效。I/O 口仍然保持进入IDLE模式以前的状态（输入或输出）。

有2种方式可以退出 IDLE 模式：

1. 执行复位操作。所有的系统和外设控制寄存器复位至默认值，但所有数据寄存器的值保持不变。因为 CLKCON.3 和 CLKCON.4 清除为“00B”，复位自动选择慢时钟（ $f_{xx}/16$ ）。如果中断被屏蔽，则复位是退出 IDLE 模式的唯一方法。
2. 激活任意使能的中断，退出 IDLE 模式。当用中断退出 IDLE 模式，CLKCON.3 和 CLKCON.4 寄存器的值保持不变，当前选择的时钟值也不变。系统会先处理中断服务程序。中断返回后，执行紧接着 IDLE 指令的下一条指令。

# 9 I/O 口

## 9.1 概述

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 有8个位可编程的 I/O 口, P0–P6。其中, P1 是7位口, P3 口是6位口, P7 口是 4位口,其它为8 位口。一共有65个I/O管脚。每个口都可以灵活配置以满足应用的需要。CPU通过直接读写口寄存器访问 I/O口。不需要特别的I/O口指令。

[表 9-1](#) 给出了S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X I/O 口的功能概述。

**表 9-1 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 口配置描述**

I/O 口	设置选项
0	可对该口的每一位进行功能设定, 可以设定为施密特触发器输入、推挽式输出或开漏输出。通过软件设定上拉电阻。 P0.0 - P0.7 也可用作外部中断 INT0–INT7 输入(噪声滤波, 中断使能和中断标志控制)
1	可对该口的每一位进行功能设定, 可以设定为施密特触发器输入、推挽式输出或开漏输出。通过软件设定上拉电阻。 P1.0–P1.6 也可以用作T1CAP, T1CLK, T1OUT, T1PWM, BUZ, SO, SCK, SI.
2	可对该口的每一位进行功能设定, 可以设定为输入或推挽式输出。通过软件设定上拉电阻。 P2.7 也可以用作AD0–AD7/VBLDREF。
3	可对该口的每一位进行功能设定, 可以设定为输入或推挽式输出。通过软件设定上拉电阻。 P3.0–P3.5 可以用作TBPWM, TAOUT/TAPWM, TACLK, TACAP, TxD, RxD 或 LCD SEG。
4	可对该口的每一位进行功能设定, 可以设定为输入、推挽式输出或开漏输出。通过软件设定上拉电阻。 P4.0–P4.7也可以用作LCD SEG信号。
5	可对该口的每一位进行功能设定, 可以设定为输入、推挽式输出或开漏输出。通过软件设定上拉电阻。 P5.0–P5.7也可以用作LCD SEG信号。
6	可对该口的每一位进行功能设定, 可以设定为输入或推挽式输出。通过软件设定上拉电阻 P6.0–P6.7也可以用作LCD SEG信号。
7	可对该口的每一位进行功能设定, 可以设定为输入或推挽式输出。通过软件设定上拉电阻 P7.0–P7.3也可以用作LCD SEG信号。
8	可对该口的每一位/两位/四位进行功能设定, 可以设定为输入、推挽式输出或开漏输出。通过软件设定上拉电阻。 P8.0–P8.7也可以用作LCD COM/SEG信号。

### 9.1.1 口数据寄存器

[表 9-2](#) 总结了S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X所有I/O 口数据寄存器的位置。P0, 1, 2, 3, 4, 5, 6, 7 和 8口的数据寄存器大致格式如[图 9-1](#) 所示。

**表 9-2 端口数据寄存器**

寄存器名称	助记标号	十进制	十六进制	位置	R/W
P0 口数据寄存器	P0	240	F0H	Set 1, Bank 1	R/W
P1 口数据寄存器	P1	241	F1H	Set 1, Bank 1	R/W
P2 口数据寄存器	P2	242	F2H	Set 1, Bank 1	R/W
P3 口数据寄存器	P3	243	F3H	Set 1, Bank 1	R/W
P4 口数据寄存器	P4	244	F4H	Set 1, Bank 1	R/W
P5 口数据寄存器	P5	245	F5H	Set 1, Bank 1	R/W
P6 口数据寄存器	P6	246	F6H	Set 1, Bank 1	R/W
P7 口数据寄存器	P7	247	F7H	Set 1, Bank 1	R/W
P8 口数据寄存器	P8	248	F8H	Set 1, Bank 1	R/W

## 9.2 P0 口

P0 口是一个8位的I/O口。P0 口设置有两个功能：

- 通用 I/O口
- 外部中断INT0–INT7的输入

可通过直接读写P0口数据寄存器P0 (地址：F0H, set 1, bank 1) 来访问P0口。

### 9.2.1 P0 口控制寄存器 (P0CONH, P0CONL)

P0 有2个8位控制寄存器：控制P0.4-P0.7的P0CONH寄存器和控制P0.0-P0.3的P0CONL寄存器。复位后，P0CONH 和 P0CONL 寄存器被清除为 "00H"，设置P0口的所有管脚为输入模式，输入模式有三种可选的配置：

- 施密特输入，下降沿触发中断.
- 施密特输入，上升沿触发中断.
- 施密特输入，上升沿/下降沿触发中断.

### 9.2.2 P0 口中断使能和标志位寄存器 (P0INTH, P0INTL)

需要将P0 作为外部中断来使用时，需要设置另外三个外部中断控制寄存器：中断使能寄存器 P0INTH (高字节, E2H, set 1, bank 1), P0INTL (低字节, E3H, set1, bank1) 和 中断标志寄存器外部中断标志寄存器P0PND (E4H, set 1, bank 1)。

中断标志寄存器P0PND可以用来在软件中检查中断标志位和清除中断标志。应用程序可以周期性的查询中断标志位P0PND来检测中断请求。

当使能了P0口的任一个管脚的中断时，则该管脚上的电平的上升沿或下降沿就会产生中断。相应的中断标志位被置为"1"，同时相应的中断级会给CPU发出中断请求。当CPU响应中断请求后，应用程序必须在软件中给相应的中断标志位P0PND位写"0"来清除中断标志。

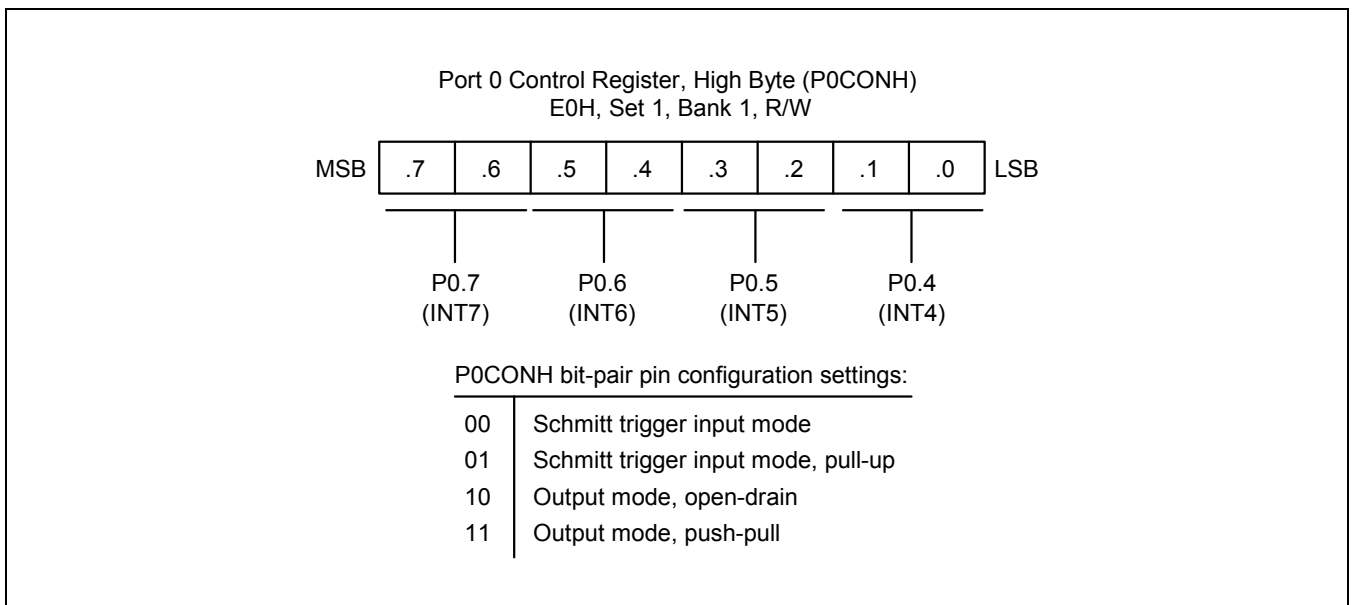


图 9-1 P0 口控制寄存器高字节 (P0CONH)

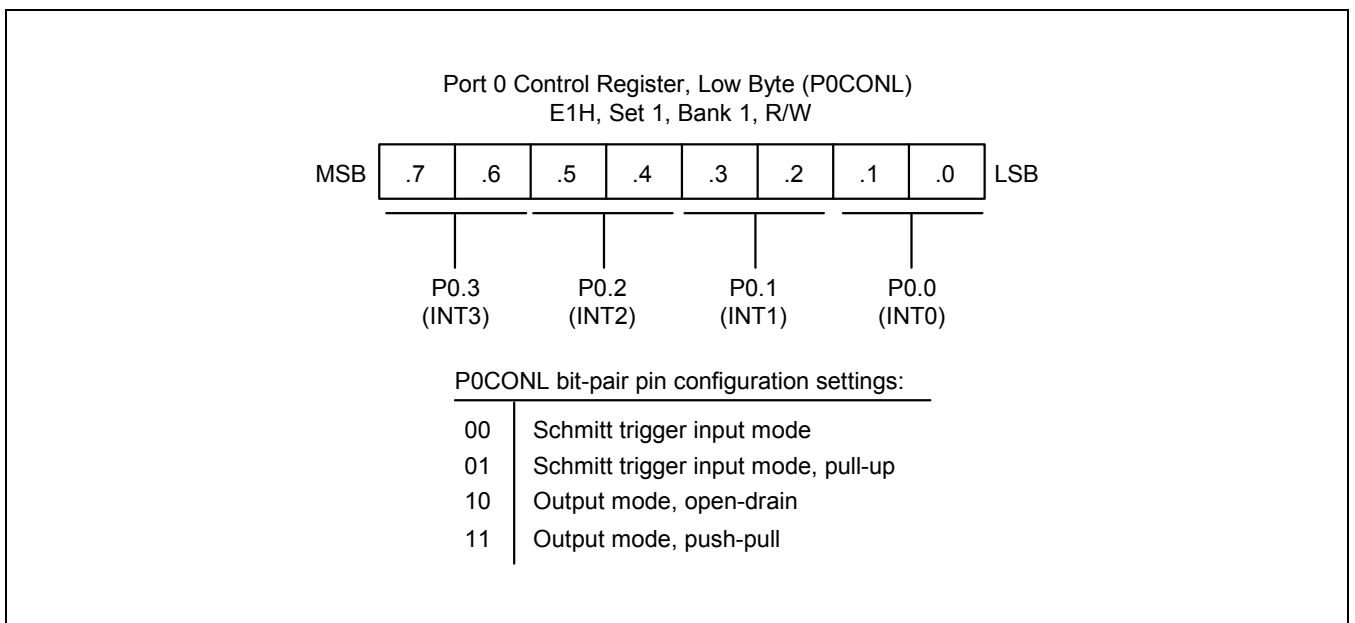


图 9-2 P0 口控制寄存器低字节 (P0CONL)



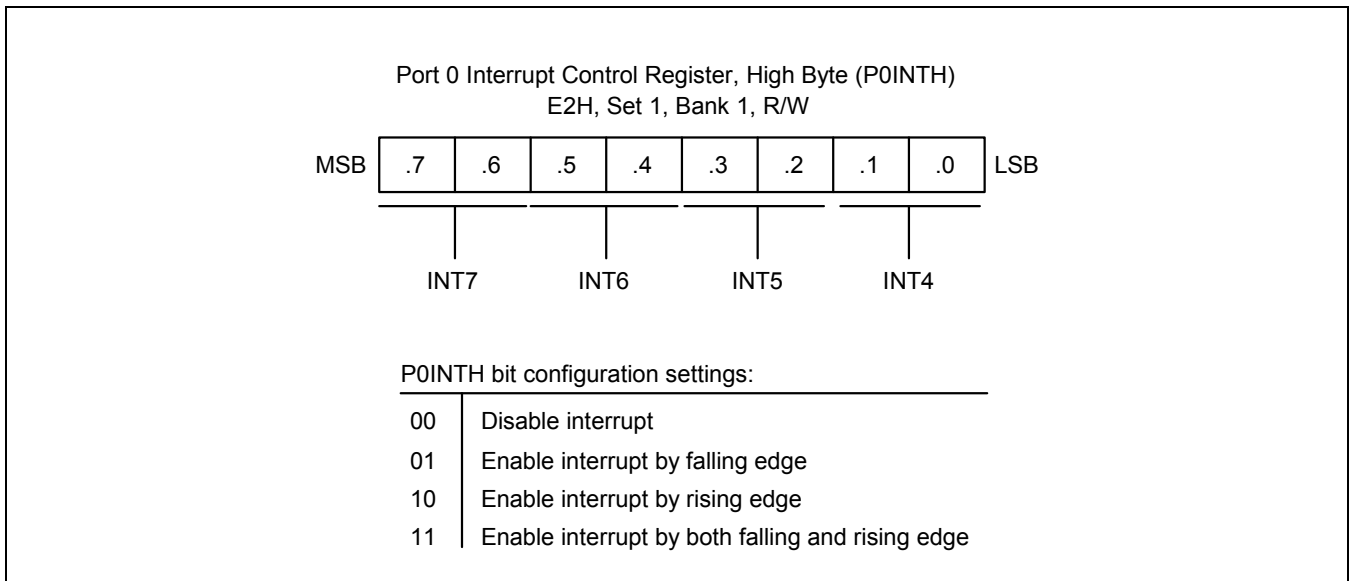


图 9-3 P0 口中断控制寄存器高字节 (POINTH)

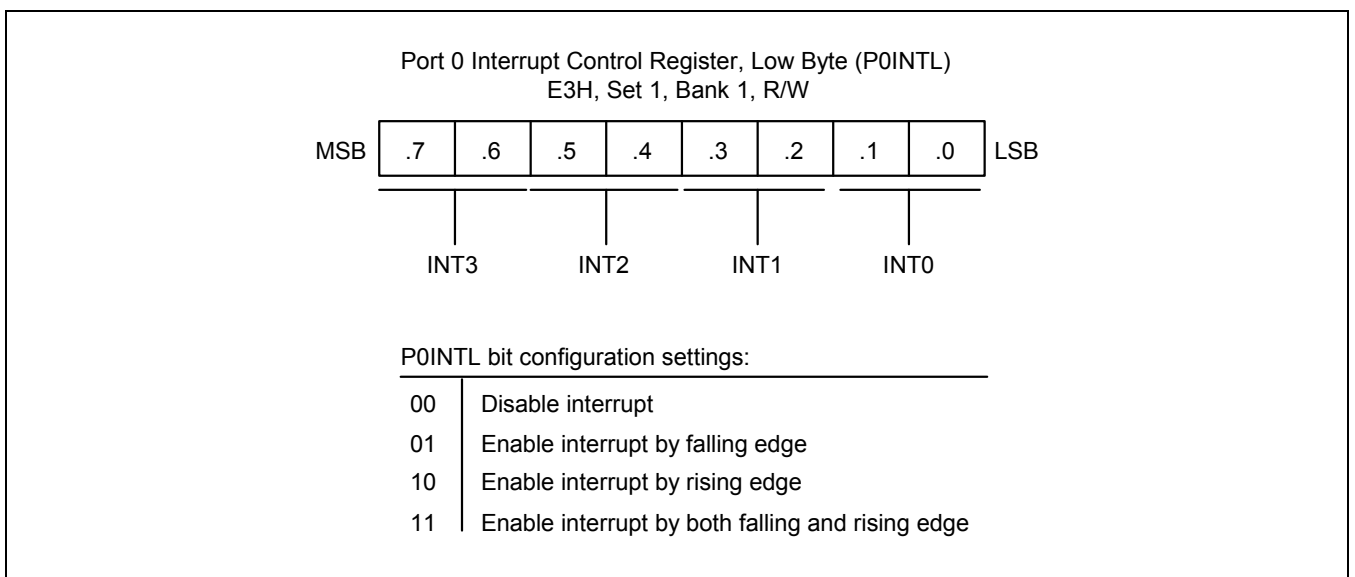


图 9-4 P0 口中断控制寄存器低字节 (POINTL)

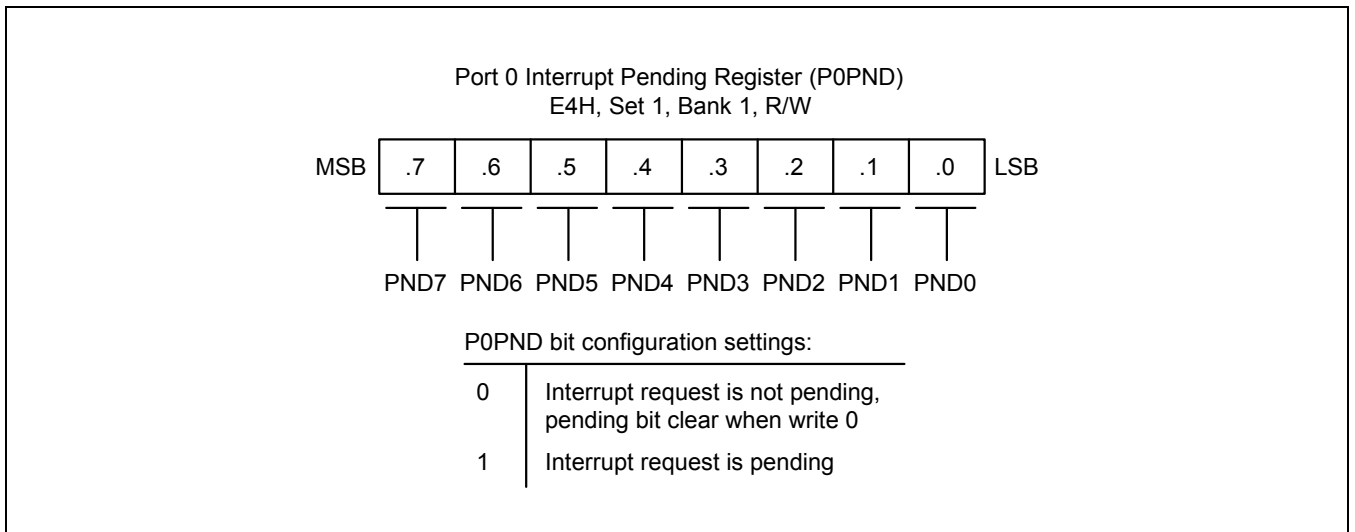


图 9-5 P0 中断标志寄存器 (P0PND)

### 9.3 P1 口

P1 口是一个7位都可单独配置的I/O口。可通过直接读写P1口数据寄存器P0 (地址: F1H set 1, bank 1) 来访问P1口。P1.0–P1.6 可用作输入(上拉/无上拉), 输出(推挽或开漏)或设置成如下的复用功能:

- 低字节(P1.0 – P1.3): T1CAP, T1CLK, T1OUT, T1PWM, BUZ
- 高字节(P1.4 – P1.6): SO, SCK, SI

#### 9.3.1 P1 口控制寄存器 (P1CONH, P1CONL)

P1 有2个8位控制寄存器: 控制P1.4 – P1.6的P1CONH寄存器和控制P1.0 – P1.3的P1CONL寄存器。复位后, P1CONH和 P1CONL寄存器被清除为 "00H", 设置P1口的所有管脚为输入模式。可以通过控制寄存器来设置各个管脚为输入或输出(推挽输出或开漏输出)或其复用功能。

对该口编程时, 请记住在P1口控制寄存器中设置外设I/O 复用功能的同时, 必须在相应的外设模块中使能该功能。

#### 9.3.2 P1 口上拉电阻使能寄存器 (P1PUR)

使用P1 口上拉电阻使能寄存器 P1PUR (E7H, set 1, bank 1)可以设置P1口各管脚的上拉电阻。

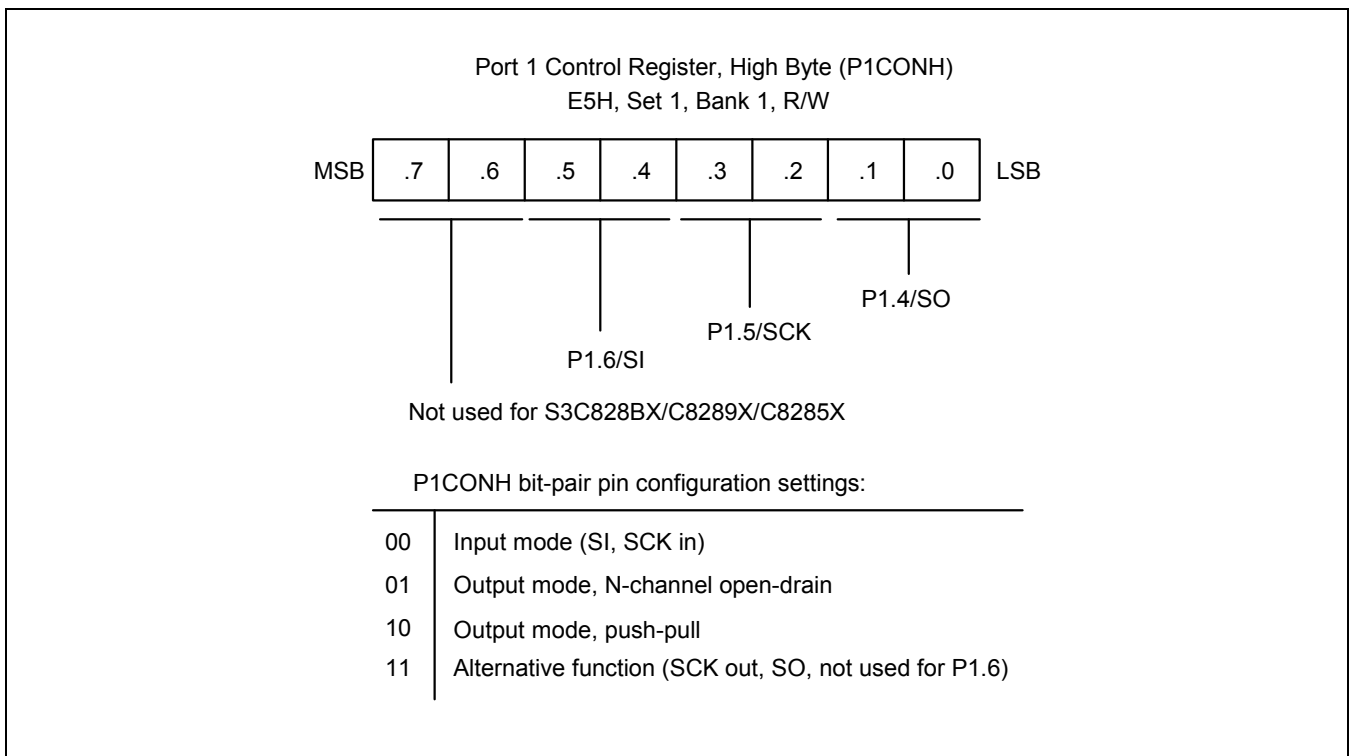


图 9-6 P1 控制寄存器高字节 (P1CONH)

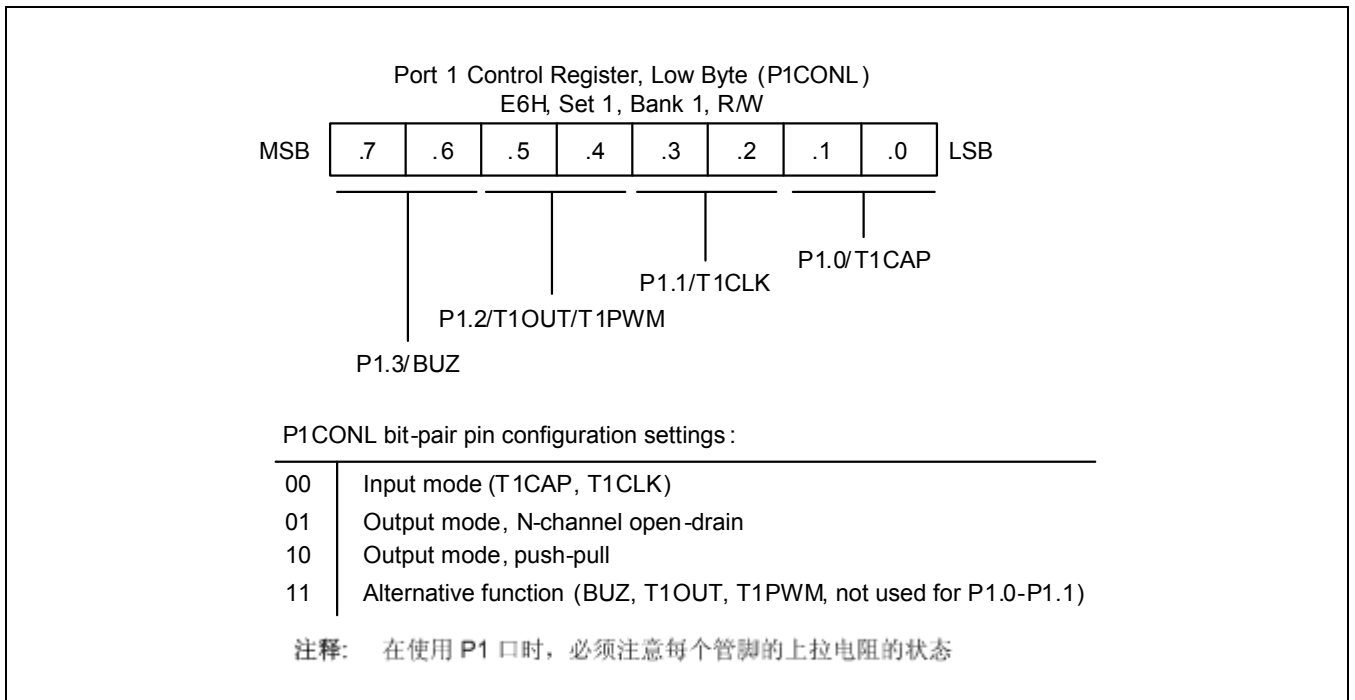


图 9-7 P1 口控制寄存器低字节 (P1CONL)

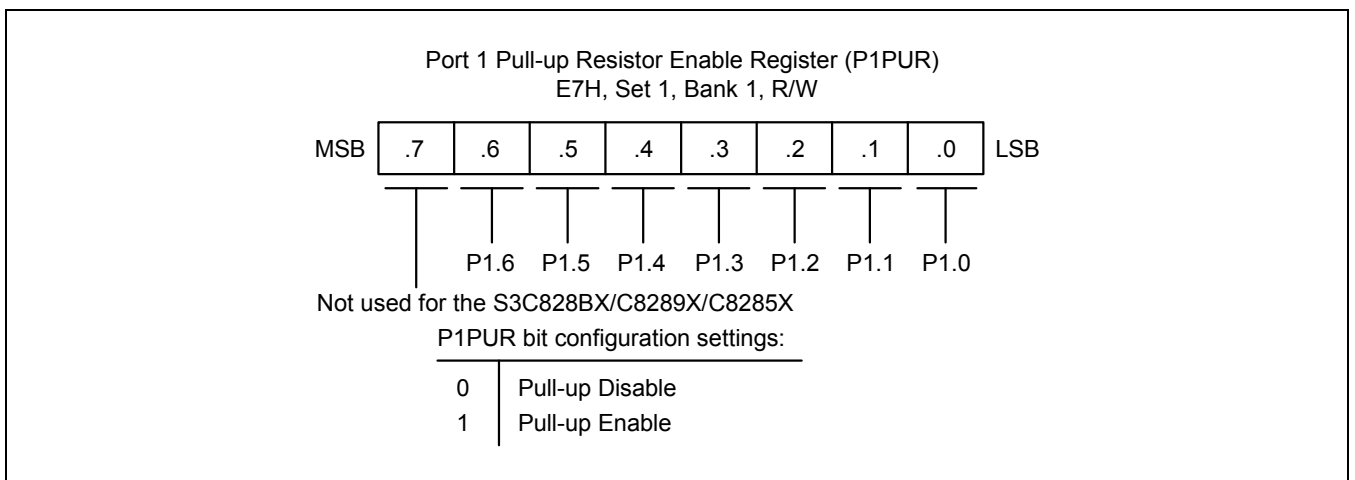


图 9-8 P1 上拉电阻使能寄存器 (P1PUR)

## 9.4 P2 口

P2 口是一个8位I/O口，可用作通用I/O口或A/D转换输入口，ADC0–ADC7。可通过直接读写P2口数据寄存器P2 (地址：F2H set 1, bank 1) 来访问P2口。

P2.0 – P2.7 可用作输入、推挽输出或设置成如下的复用功能。在输入模式下，也可用作ADC输入或外部参考电压输入：

- 低字节(P2.0–P2.3): AD0–AD3
- 高字节(P2.4–P2.7): AD4–AD7,  $V_{BLDREF}$

### 9.4.1 P2 口控制寄存器 (P2CONH, P2CONL)

P2 有2个8位控制寄存器：控制P2.4 – P2.7的P2CONH寄存器和控制P2.0 – P2.3的P2CONL寄存器。复位后，P2CONH和 P2CONL寄存器被清除为 "00H"，设置P2口的所有管脚为输入模式。可以通过控制寄存器来设置各个管脚为输入、推挽输出或其复用功能。

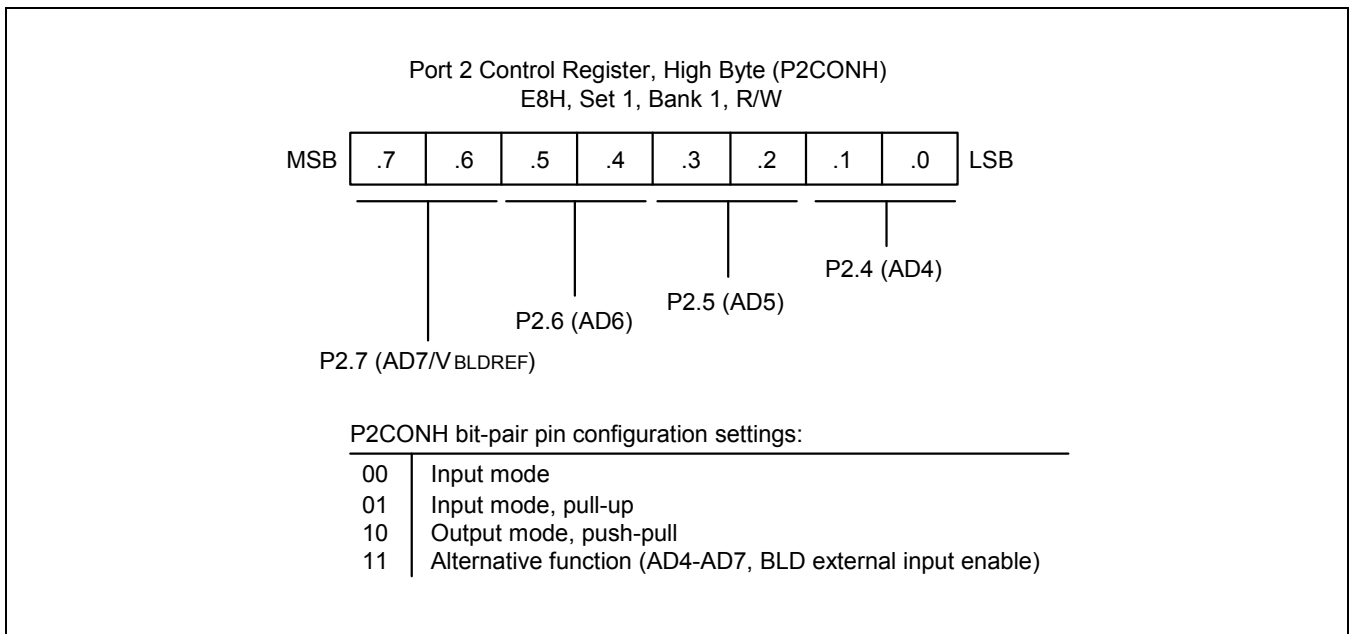


图 9-9 P2 口控制寄存器高字节 (P2CONH)

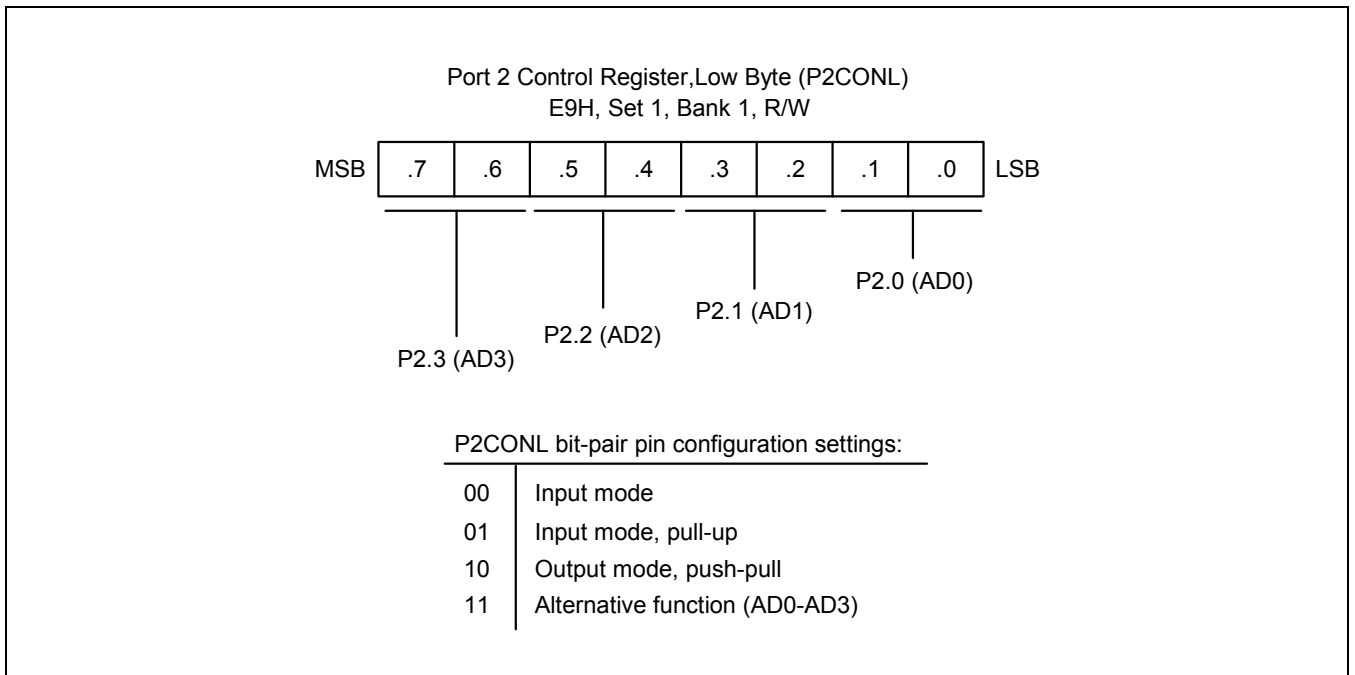


图 9-10 P2 口控制寄存器低字节 (P2CONL)

## 9.5 P3 口

P3 口是一个6位都可单独配置的I/O口。可通过直接读写P3口数据寄存器P3 (地址: F3H set 1, bank 1) 来访问P3口。P3.0–P3.5 可用作输入(上拉/无上拉)或推挽输出, P3.0 – P3.3可设置为LCD SEG 信号或设置成如下的复用功能:

- 低字节(P3.0–P3.3): TBPWM, TAOUT, TAPWM, TACLK, TACAP
- 高字节(P3.4–P3.6): TxD, RxD

### 9.5.1 P3 口控制寄存器 (P3CONH, P3CONL)

P3 有2个8位控制寄存器: 控制P3.4 – P3.5的P3CONH寄存器和控制P3.0 – P3.3的P3CONL寄存器。复位后, P3CONH和 P3CONL寄存器被清除为 "00H", 设置所有管脚为输入模式。可以通过控制寄存器来设置各个管脚为输入或输出, 使能上拉电阻或使其复用功能。

对该口编程时, 请记住在P3口控制寄存器中设置外设I/O 复用功能的同时, 必须在相应的外设模块中使能该功能。

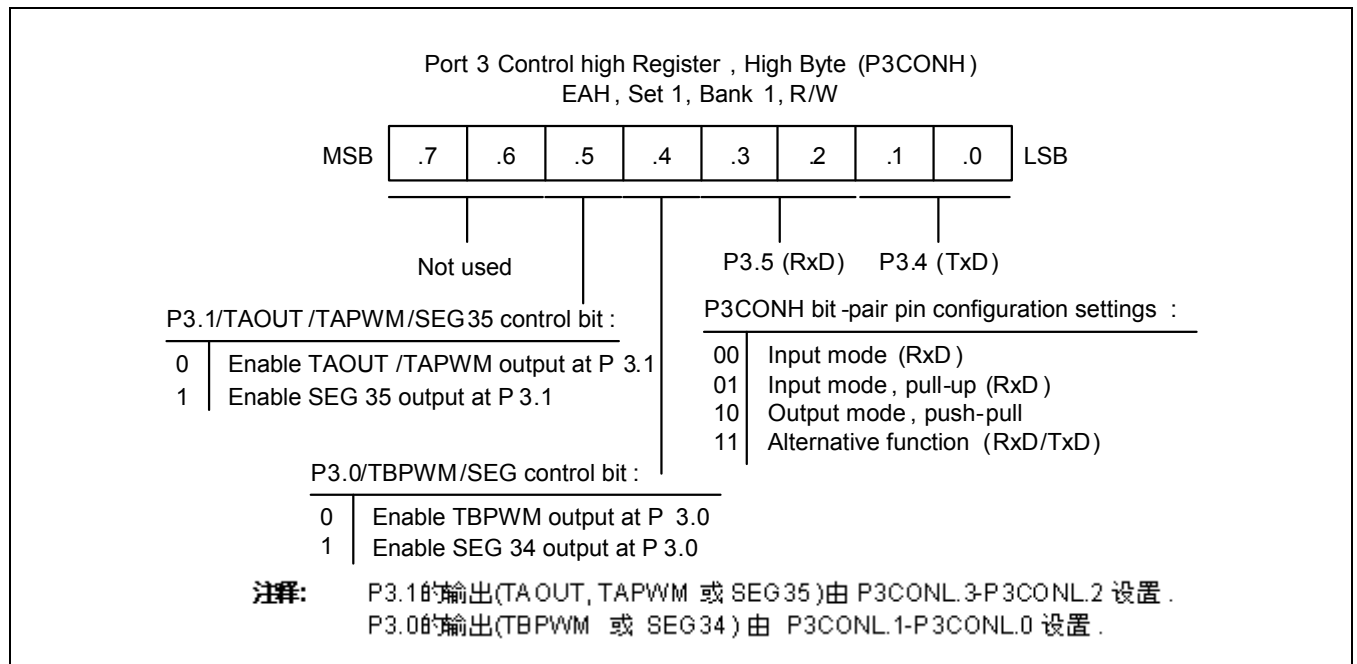


图 9-11 P3 口控制寄存器高字节 (P3CONH)

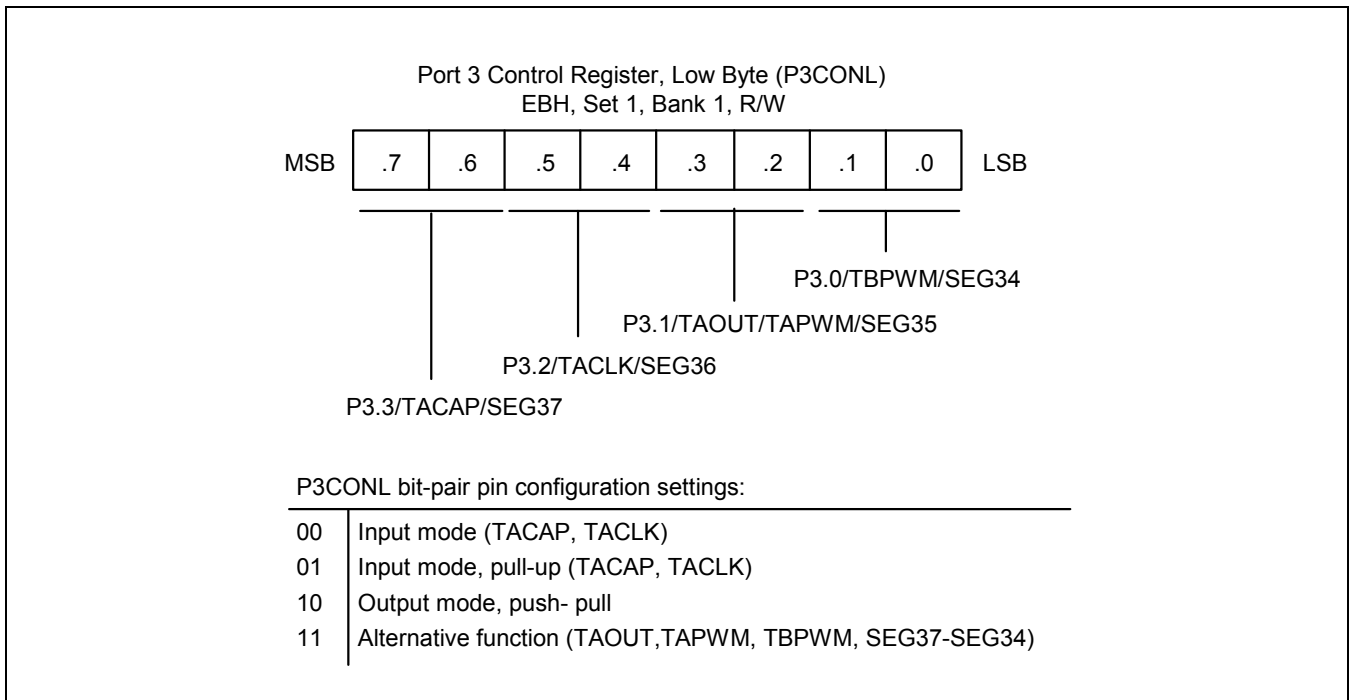


图 9-12 P3 口控制寄存器低字节 (P3CONL)



## 9.6 P4 口

P4 口是一个8位都可单独配置的I/O口。可通过直接读写P4口数据寄存器P4 (地址: F4H set 1, bank 1) 来访问P4口。P4.0–P4.7 可用作输入(上拉/无上拉)、输出(开漏或推挽), 或可用作LCD SEG信号:

### 9.6.1 P4 口控制寄存器 (P4CONH, P4CONL)

P4有2个8位控制寄存器: 控制P4.4 – P4.7的P4CONH寄存器和控制P4.0 – P4.3的P4CONL寄存器。复位后, P4CONH和 P4CONL寄存器被清除为 "00H", 设置所有管脚为输入模式。

### 9.6.2 P4 口上拉电阻使能寄存器 (P4PUR)

使用P4 口上拉电阻使能寄存器 P4PUR (EEH, set 1, bank 1)可以设置P4口各管脚的上拉电阻。

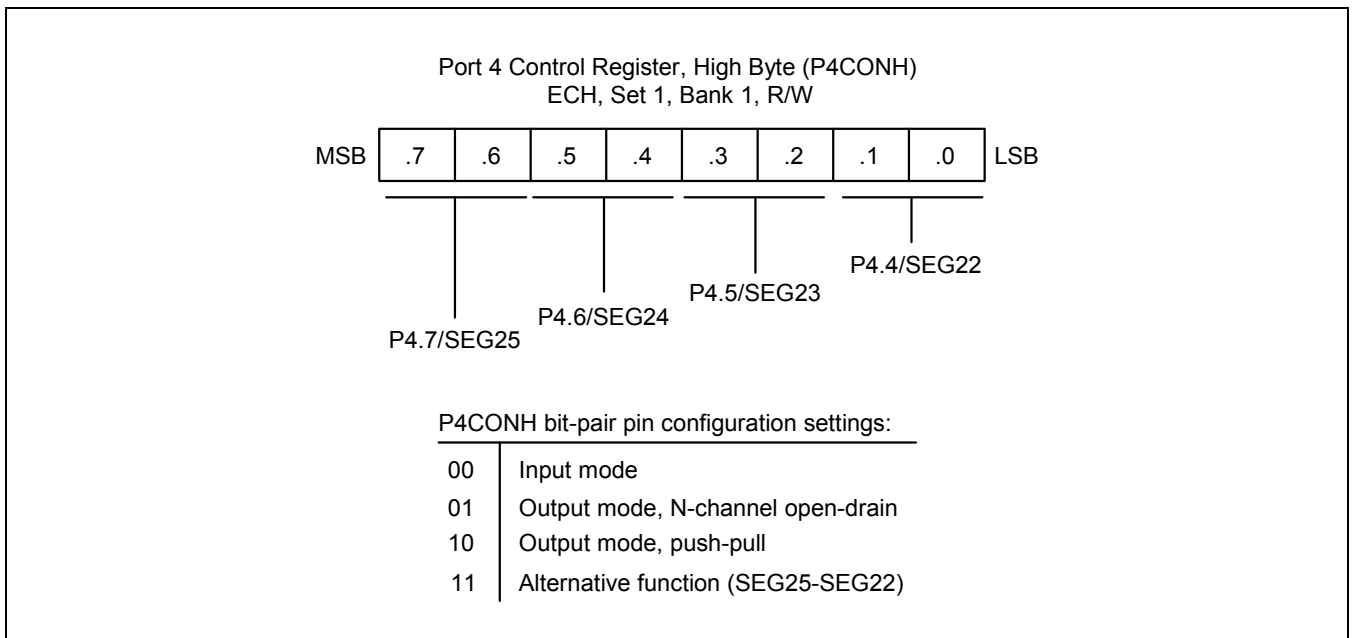


图 9-13 P4 口控制寄存器高字节 (P4CONH)

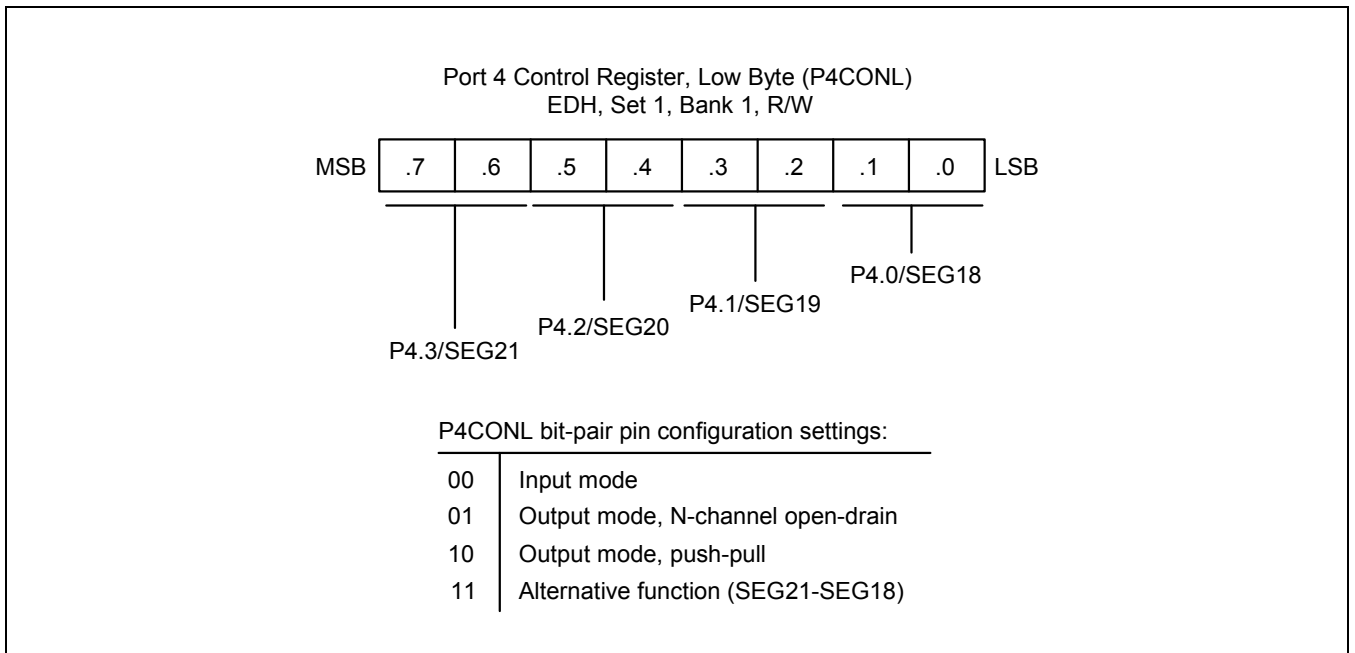


图 9-14 P4 口控制寄存器低字节 (P4CONL)

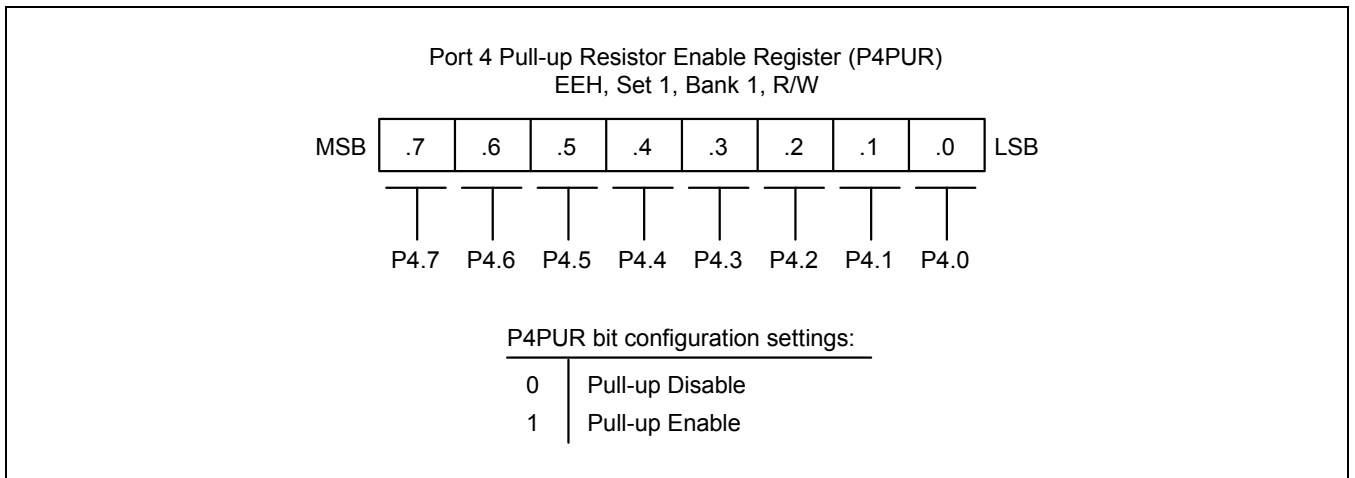


图 9-15 P4 上拉电阻使能寄存器 (P4PUR)

## 9.7 P5 口

P5 口是一个8位都可单独配置的I/O口。可通过直接读写P5口数据寄存器P5 (地址: F5H set 1, bank 1) 来访问P5口。P5.0 – P5.7 可用作输入(上拉/无上拉)、输出(开漏或推挽), 或可用作LCD SEG信号:

### 9.7.1 P5 口控制寄存器 (P5CONH, P5CONL)

P5有2个8位控制寄存器: 控制P5.4 – P5.7的P5CONH寄存器和控制P5.0 – P5.3的P5CONL寄存器。复位后, P5CONH和 P5CONL寄存器被清除为 "00H", 设置所有管脚为输入模式。

### 9.7.2 P5 口上拉电阻使能寄存器 (P5PUR)

使用P5 口上拉电阻使能寄存器 P5PUR (EFH, set 1, bank 1)可以设置P5口各管脚的上拉电阻。

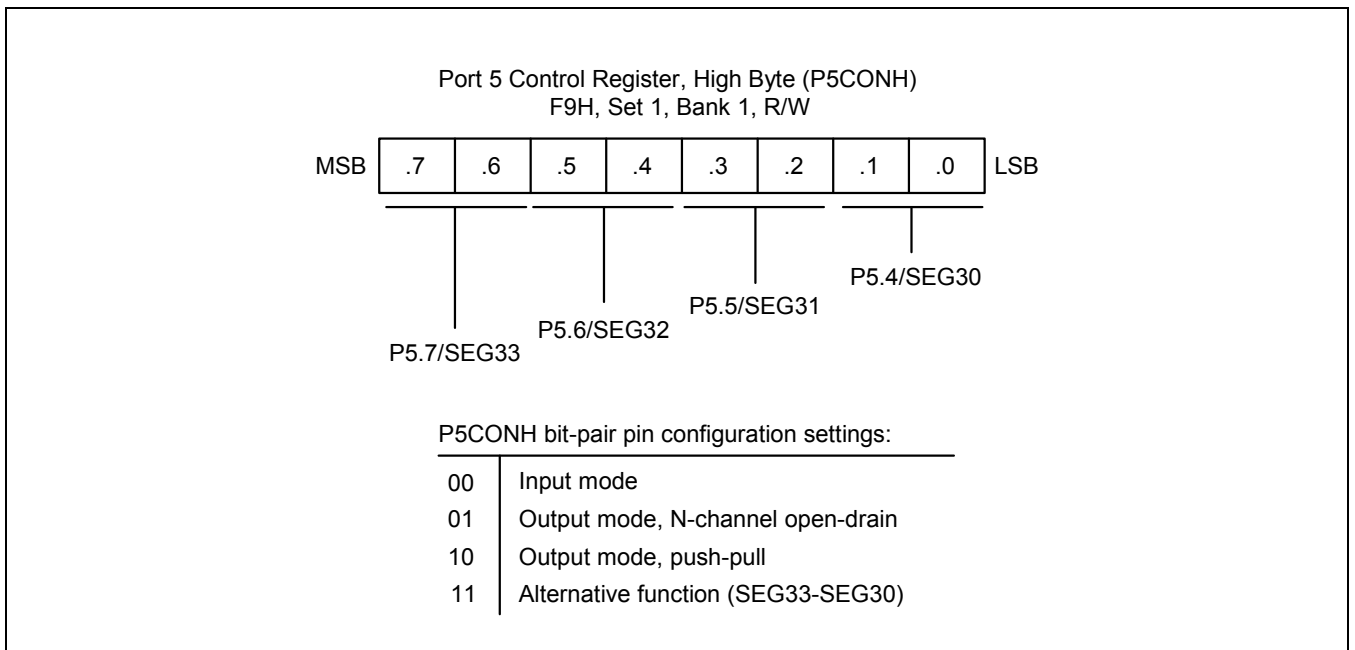


图 9-16 P5 口控制寄存器 (P5CONH)

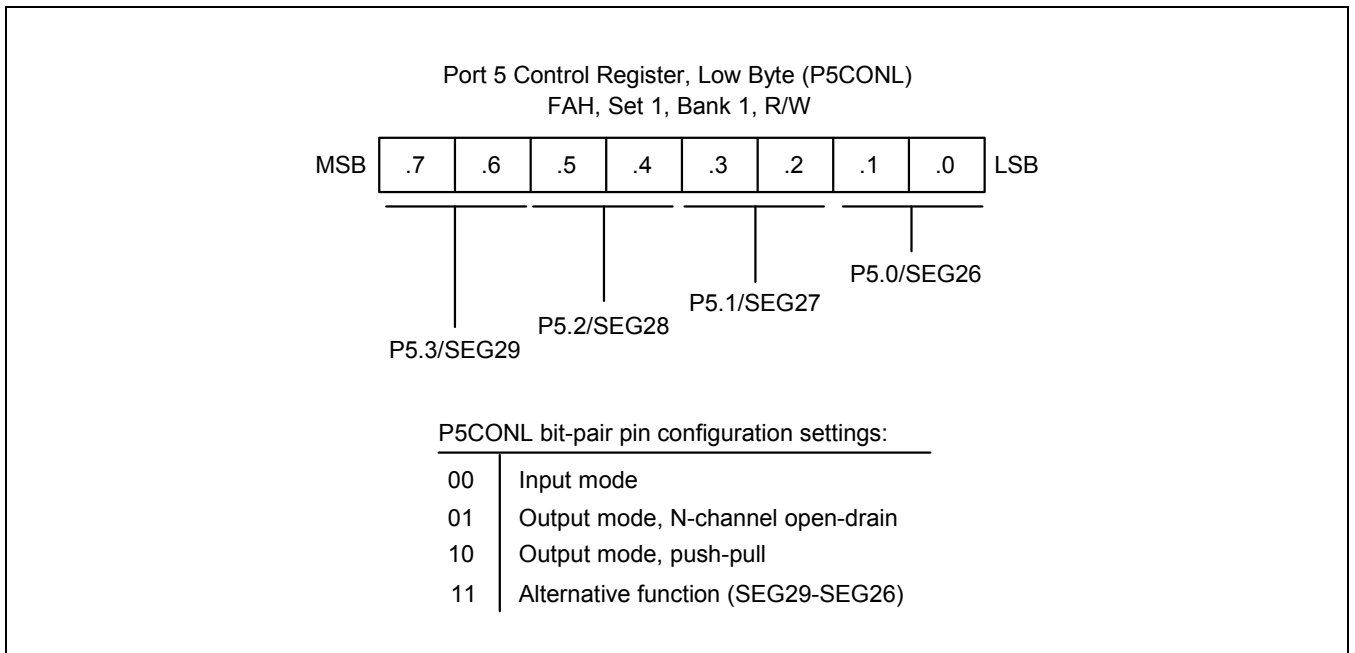


图 9-17 P5 口控制寄存器低字节 (P5CONL)

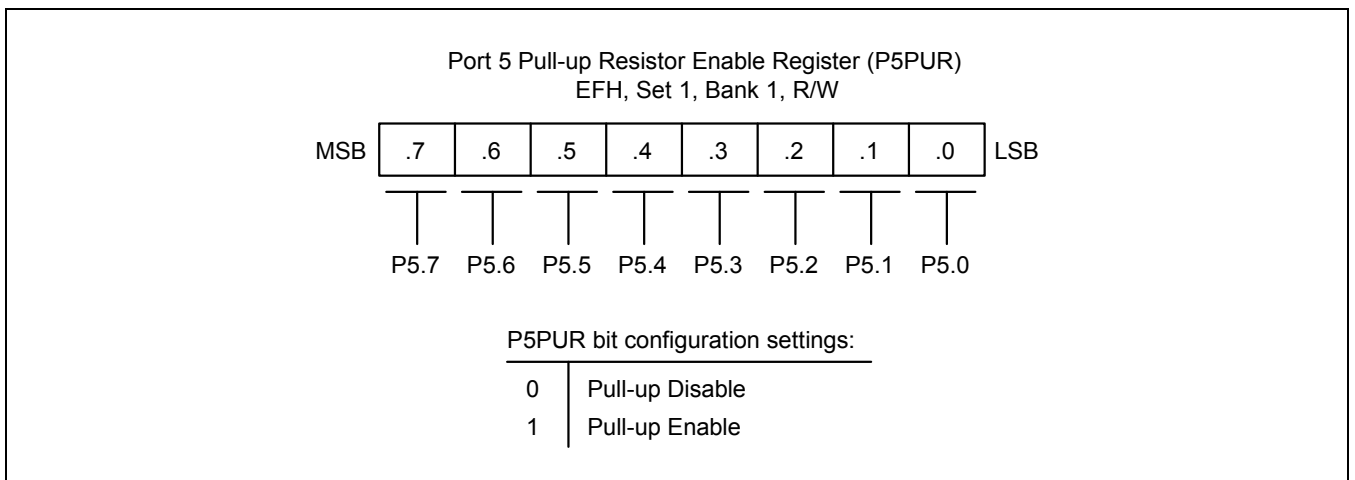


图 9-18 P5 口上拉电阻使能寄存器 (P5PUR)

## 9.8 P6 口

P6 口是一个8位都可单独配置的I/O口。可通过直接读写P6口数据寄存器P6 (地址: F6H set 1, bank 1) 来访问P6口。P6.0 – P6.7 可用作输入(上拉/无上拉)、推挽输出, 或可用作LCD SEG信号:

### 9.8.1 P6 口控制寄存器 (P6CONH, P6CONL)

P6有2个8位控制寄存器: 控制P6.4 – P6.7的P6CONH寄存器和控制P6.0 – P6.3的P6CONL寄存器。复位后, P6CONH和 P6CONL寄存器被清除为 "00H", 设置所有管脚为输入模式。

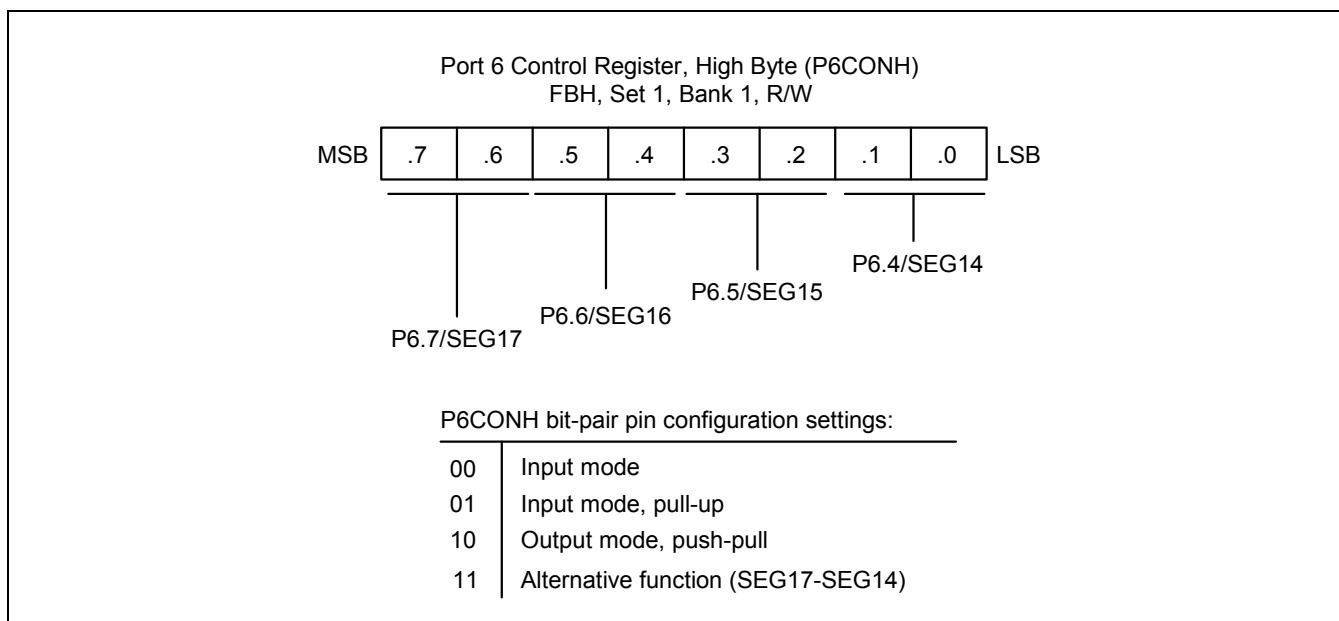


图 9-19 P6 口控制寄存器高字节 (P6CONH)

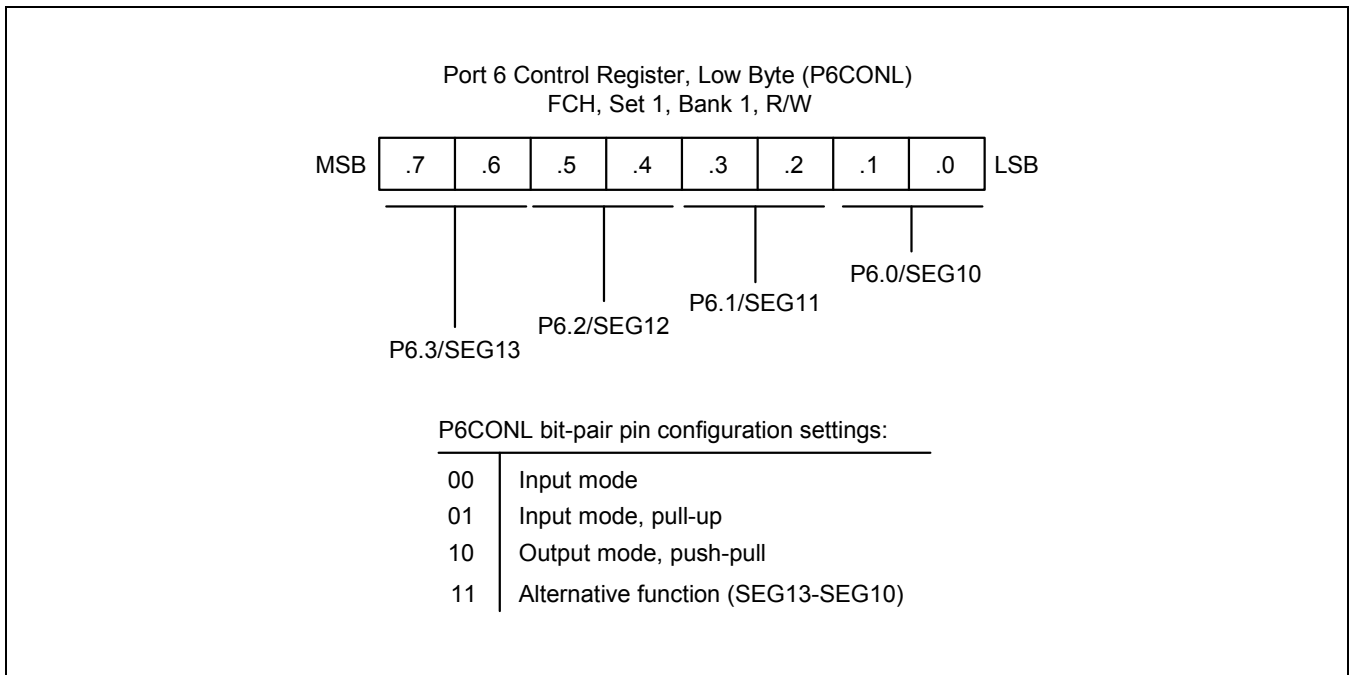


图 9-20 P6 控制寄存器低字节 (P6CONL)

## 9.9 P7 口

P7 口是一个4位都可单独配置的I/O口。可通过直接读写P7口数据寄存器P7 (地址: F7H set 1, bank 1) 来访问P7口。P7.0–P7.3 可用作输入(上拉/无上拉)、推挽输出, 或可用作LCD SEG信号:

### 9.9.1 P7 口控制寄存器 (P7CON)

P7有1个8位控制寄存器: 控制P7.0 – P7.3的P7CON寄存器。复位后, P7CON寄存器被清除为 "00H", 设置所有管脚为输入模式。

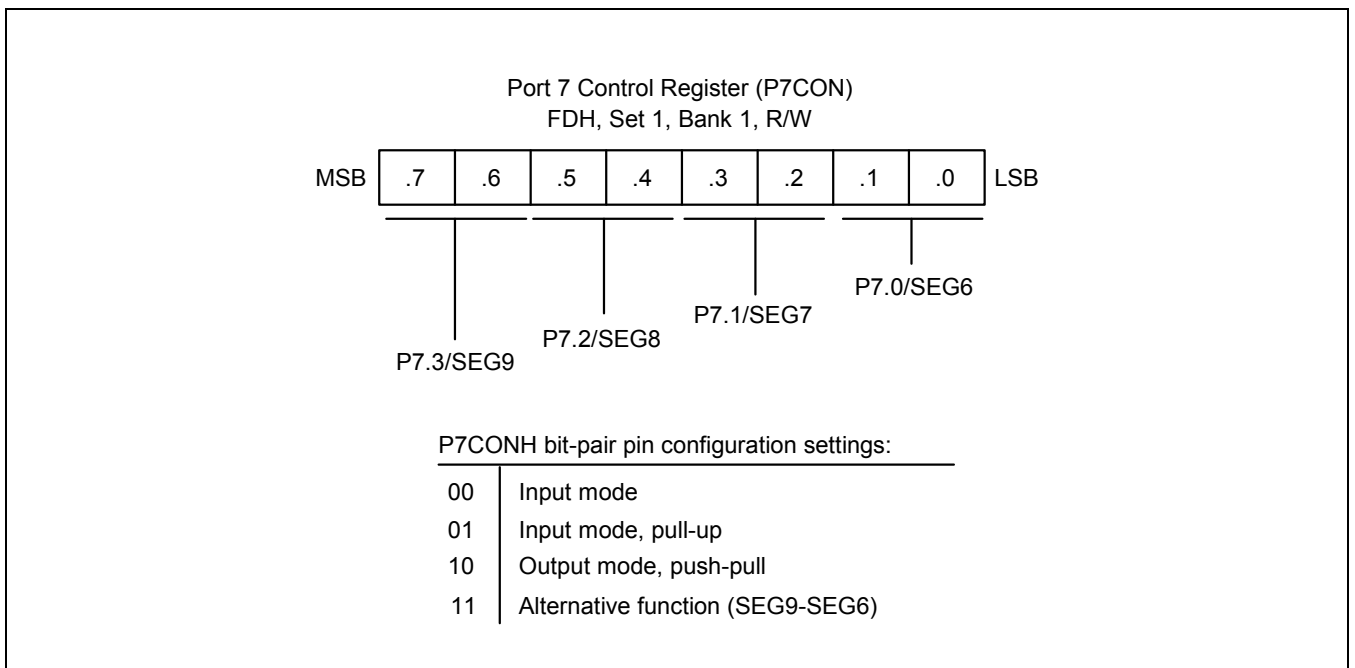


图 9-21 P7 口控制寄存器 (P7CON)

## 9.10 P8 口

P8 口是一个8位都可单独配置的I/O口。可通过直接读写P8口数据寄存器P8 (地址: F8H set 1, bank 1) 来访问P8口。P8.0 – P8.7 可用作输入(上拉/无上拉)、推挽输出, 或可用作LCD SEG信号:

### 9.10.1 P8 口控制寄存器 (P8CON)

P8有1个8位控制寄存器: 控制P8.0 – P8.7的P8CON寄存器。复位后, P8CON寄存器被清除为 "00H", 设置所有管脚为输入模式。

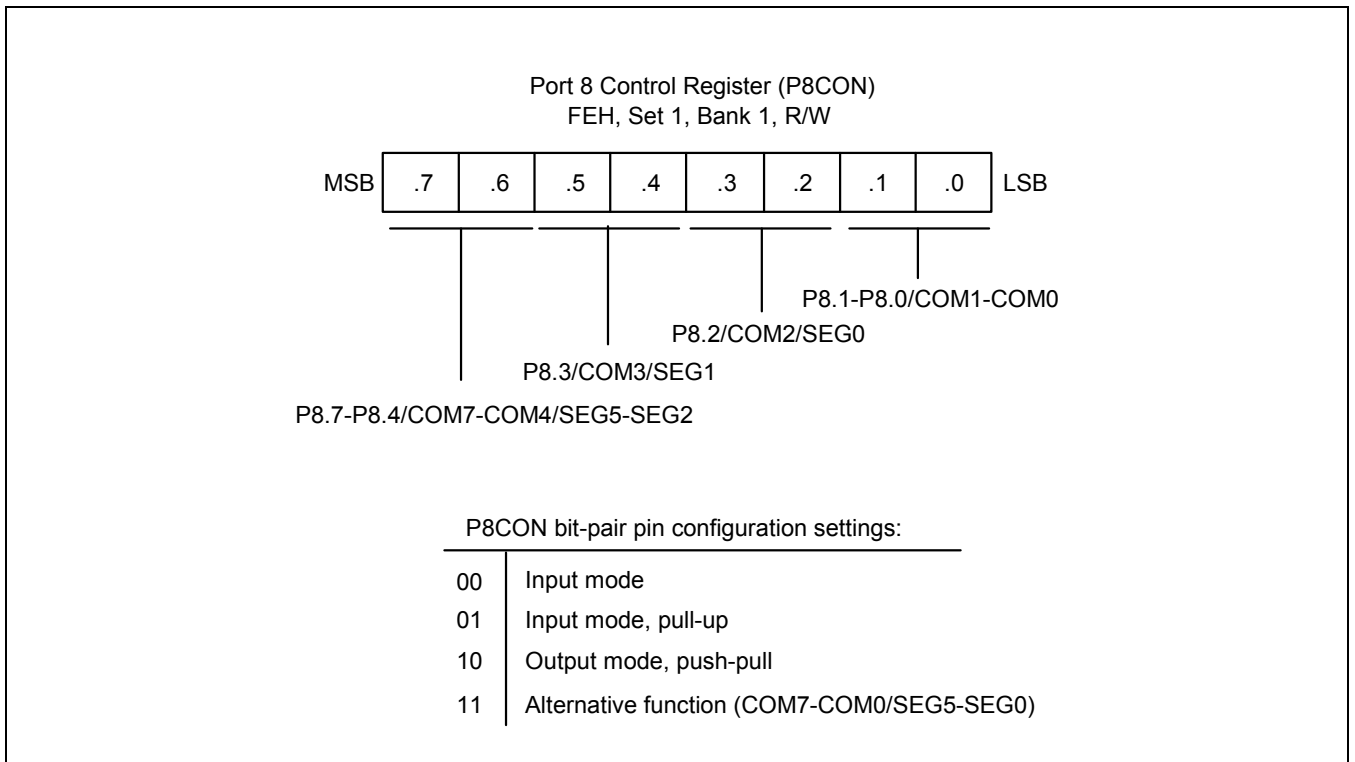


图 9-22 P8 口控制寄存器 (P8CON)



# 10

## BASIC TIMER

### 10.1 概述

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 内置一个8位Basic Timer。

#### 10.1.1 BASIC TIMER (BT)

Basic Timer 主要应用在两个方面：

- 当系统出现异常时，Basic Timer 作为看门狗定时器，自动复位芯片。
- 在复位或退出 STOP 模式后，用于延时以稳定振荡。

Basic Timer 单元包含以下几个部分：

- 时钟分频器 (fosc/4096/1024/128/16)
- 8 位 Basic Timer 计数器 BTCNT (地址: FDH, Set 1, Bank 0, 只读)
- Basic Timer 控制寄存器 BTCON (地址: D3H, Set 1, 可读/写)

### 10.1.2 BASIC TIMER 控制寄存器 (BTCON)

Basic Timer 控制寄存器 BTCON，用于选择输入时钟频率、清除 BT 计数器和分频器、禁止或使能看门狗功能。它的物理地址位于D3H，set 1，可通过寄存器寻址模式进行读/写。

复位后，BTCON 被清零“00H”，该选项将使能看门狗功能，并选择 BT 的时钟为 fxx/4096。如果想禁用看门狗功能，必须将 Basic Timer 控制寄存器的高四位 BTCON.7–BTCON.4 设置为“1010B”。

在正常操作情况下，可以写“1”到 BTCON.1 来清零 8 位 Basic Timer 计数器 BTCNT (set 1, bank 0, FDH)。也可以写“1”到BTCON.0 位，来清除 Basic Timer 输入时钟的分频器。

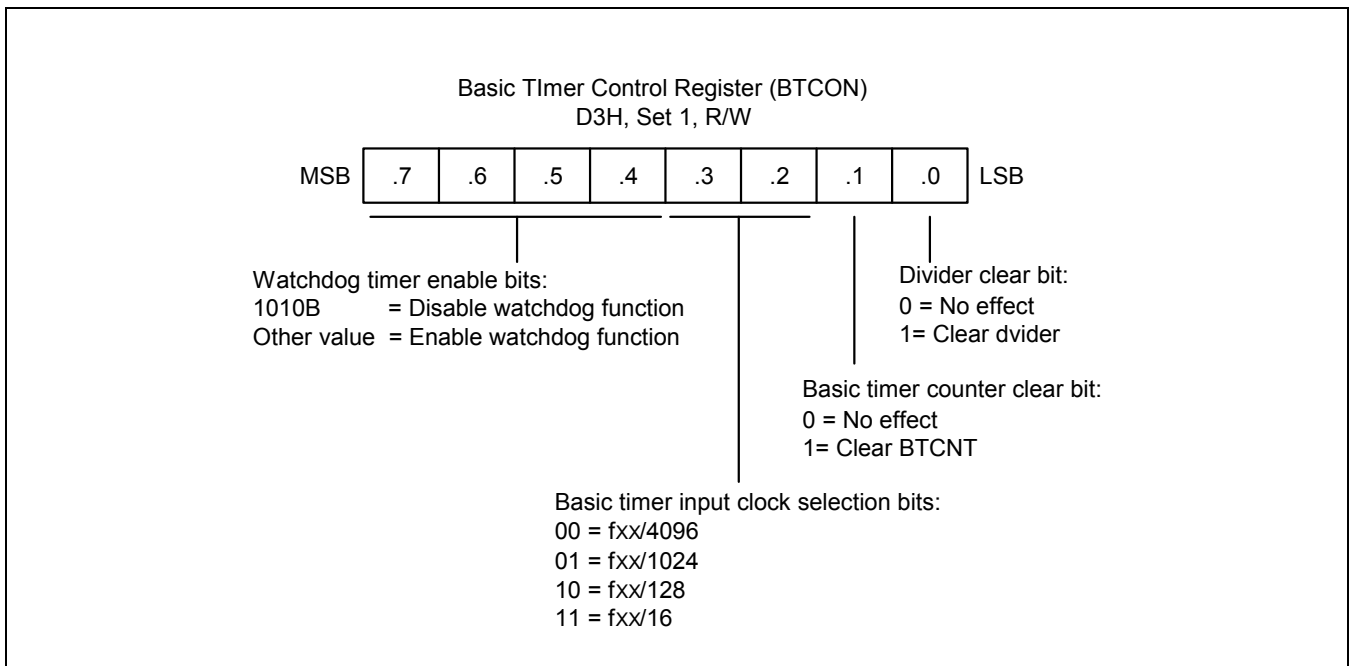


图 10-1 Basic Timer 控制寄存器 (BTCON)

### 10.1.3 BASIC TIMER 功能描述

#### 10.1.3.1 看门狗定时器功能

可通过向 BTCON.7–BTCON.4 写入非“1010B”的任意值(“1010B”将禁用看门狗功能), 来对basic timer溢出信号(BTOVF)进行编程控制, 以产生系统复位。复位时 BTCON清零, 自动允许看门狗功能。同时选择 Basic Timer 的时钟信号为 fxx/4096。

无论什么时候 BT 计数器溢出, 都将产生复位。正常情况下, 应用程序须防止计数器溢出, 为此, 每隔一段时间要清除看门狗计数器 (写“1”到 BTCON.1)。

如果由于电路噪声或其它原因造成系统失灵, Basic Timer 的清零操作不会被执行, 看门狗将复位芯片。换言之, 在正常运行期间, 必须通过 BTCNT 的清零操作来打破 Basic Timer 的溢出循环(8位basic timer 计数器BTCNT的第7位溢出)。如果发生异常, 自动产生复位。

#### 10.1.3.2 振荡稳定功能

还可以用 Basic Timer 对复位或 STOP 模式唤醒时的振荡稳定时间进行定时。

在 STOP 模式下, 每当复位或者外部中断发生, 振荡电路开始运行。然后 BTCNT 寄存器按照 fxx/4096 的频率 (复位时) 或者按照预设时钟 (对外部中断)开始计数。当 BTCNT.4 溢出时, 将产生一个信号表明振荡稳定时间到, CPU开始在稳定的时钟下开始工作。

总之, 当系统从 STOP 模式退出时, 以下的事件会依次发生:

1. 上电复位或外部中断产生, 系统从 STOP 模式唤醒, 振荡器起振。
2. 如果是复位操作, BTCNT 将以默认的时钟频率 fxx/4096 进行计数。如果是外部中断, BTCNT 寄存器按照预设时钟频率计数。
3. 振荡稳定间隔计时开始, 直到BTCNT第4位溢出为止。
4. 当BTCNT.4 溢出发生时, CPU 开始正常工作。

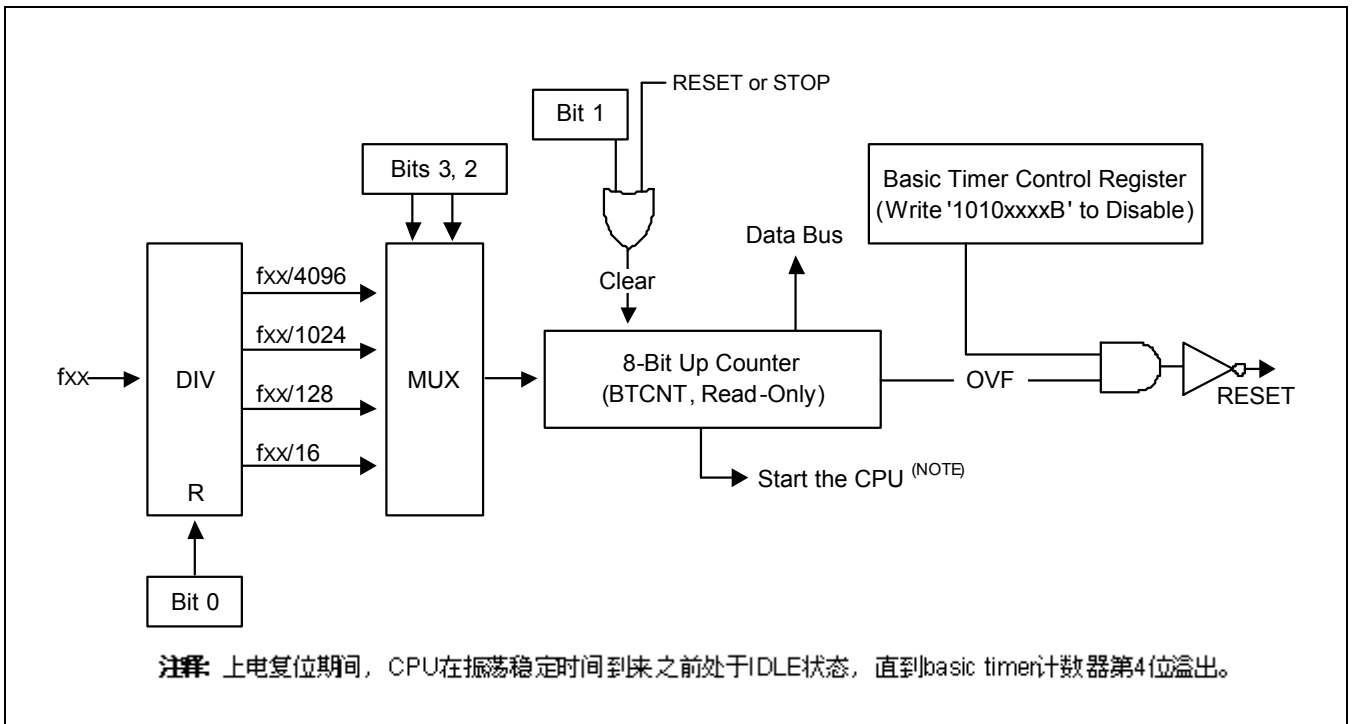


图 10-2 Basic Timer 方框图

# 11

## 8 位 TIMER A/B

### 11.1 8 位TIMER A

#### 11.1.1 概述

Timer A 是一个 8 位通用 Timer/Counter。它有三种工作模式，可通过配置寄存器TACON 来选择其中的一种：

- Interval（定时）模式 (TAOUT 反转输出)
- Capture（捕获）模式，TACAP 上升沿或下降沿触发
- PWM 模式 (TAPWM)

Timer A 包括以下几个功能模块：

- 带多路复用的时钟分频器 (fxx/1024/256/64/8/1)
- 外部时钟输入管脚 (TACLK)
- 一个 8 位计数器 (TACNT)，一个 8 位比较器和一个 8 位参考数据寄存器 (TADATA)
- I/O 管脚用于捕获输入管脚 (TACAP) 或 PWM 输出或匹配输出 (TAPWM, TAOUT)
- Timer A 溢出中断 (IRQ0, 中断向量地址: DEH) 和匹配/捕获中断 (IRQ0, 中断向量地址: DCH) 产生
- Timer A 控制寄存器 TACON (地址: E8H, Set 1, Bank 0, 可读/写)

### 11.1.2 TIMER A 控制寄存器 (TACON)

Timer A 控制寄存器 TACON 可以：

- 选择 Timer A 工作模式 (定时模式、捕获模式或 PWM 模式)
- 选择 Timer A 输入时钟频率
- 将 Timer A 计数器 TACNT 清零
- 使能 Timer A 溢出中断或 Timer A 匹配/捕获中断
- 清除 Timer A 匹配/捕获中断标志位

TACON 地址为：E8H, Set 1, Bank 0。可读/写，支持寄存器寻址模式。

复位时，将清除 TACON至“00H”，将设置 Timer A 为定时模式，选择 fxx/1024 作为输入时钟频率，并且禁止所有的 Timer A 中断。正常工作时，任何时刻都可通过往 TACON.2 位写“1”来将 Timer A 计数器清零。

Timer A 的溢出中断 (TAOVF) 的中断优先级为 IRQ0，中断向量地址为 DEH。当 Timer A 溢出中断发生后，CPU 响应此中断，中断标志位自动被硬件清零或必须由软件清零。

往 TACON.1 位写“1”可使能 Timer A 的匹配/捕获中断 (IRQ0，中断向量地址：DCH)。

为了检测到一个匹配/捕获中断标志条件，应用程序需查询 INTPND.1 位。当检测到“1”时，Timer A 的匹配或捕获中断产生。执行中断服务程序时，中断标志位必须通过软件的方法清除，即写“0”到 Timer A 的匹配/捕获中断标志位 INTPND.1。

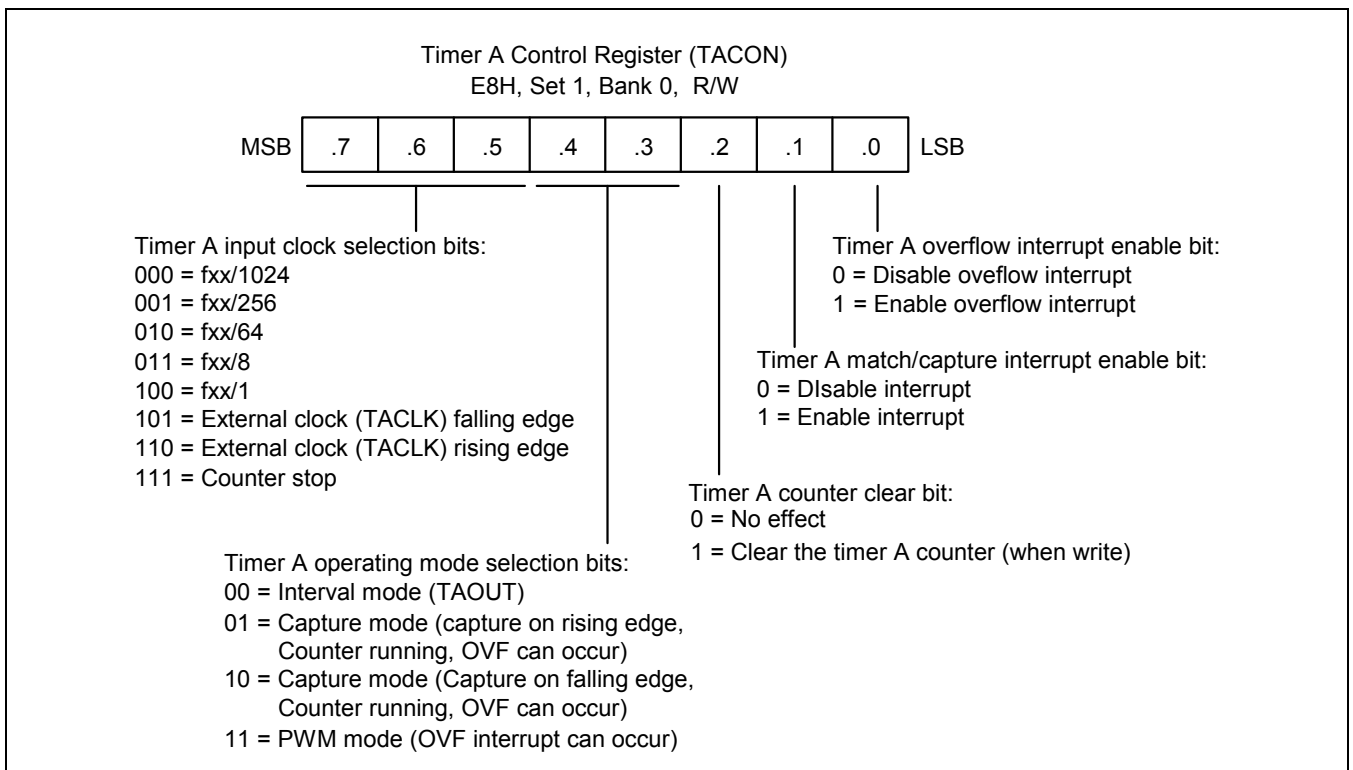


图 11-1 Timer A 控制寄存器 (TACON)

### 11.1.3 TIMER A 功能描述

#### 11.1.3.1 Timer A 中断 (IRQ0, 中断向量地址: DCH 和 DEH)

Timer A 模块能产生 2 种中断: Timer A 溢出中断 (TAOVF), 以及 Timer A 匹配/捕获中断 (TAINT)。TAOVF 的中断优先级为 IRQ0, 中断向量地址为 DEH。TAINT 的中断优先级也为 IRQ0, 但不同的是中断向量地址为 DCH。

当 Timer A 溢出中断发生后, CPU 响应此中断, 中断标志位自动被硬件清零或必须由软件清零, 即在中断服务程序中写“0”到中断标志位 INTPND.0。但对于 Timer A 匹配/捕获中断标志位来说, 必须通过在应用程序的中断服务程序中写“0”到中断标志位 INTPND.1 来清零。

#### 11.1.3.2 Interval (定时) 模式

在定时模式中, 当计数器的值与 Timer A 参考数据寄存器 TADATA 写入的值相等时, 将产生一个匹配信号。这个匹配信号产生一个 Timer A 的匹配中断(TAINT, 中断向量地址: DCH) 并将计数器清零。

举个例子, 如果写“10H”到 TADATA, 计数器将增计数直到值为“10H”。当计数器值达到“10H”时, Timer A 中断请求产生, 计数器的值复位, 之后重新计数。每一次匹配时, Timer A 的输出管脚上的信号电平将反转 (见[图 11-2](#))。

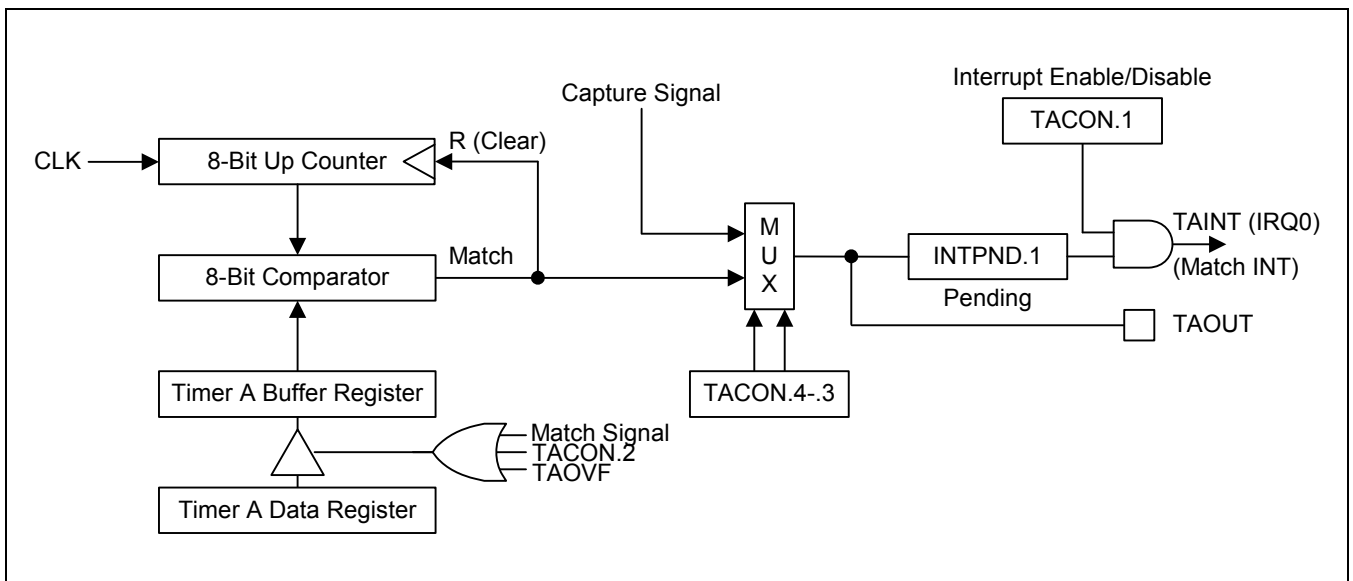


图 11-2 简化的 Timer A 功能框图: Interval (定时) 模式

### 11.1.4 PWM 模式

脉冲宽度调制 (PWM) 模块可以让用户编程控制 TAPWM 管脚上输出脉冲宽度 (持续时间)。和定时模式一样, 当计数器的值和 Timer A 数据寄存器写入的值一致时, 产生一个匹配信号。不同的是, 在 PWM 模式中, 这个匹配信号并不将计数器清零, 而是继续计数直到“FFH”溢出, 之后从“00H”重新开始向上计数。

虽然在 PWM 模式下可以使用 Timer A 的匹配信号产生一个 Timer A 溢出中断, 但这些中断在 PWM 类型的应用中并不普遍。当参考数据值小于或者等于( $\leq$ )计数器值时, TAPWM 管脚上的脉冲保持低电平; 而当参考数据值大于( $>$ )计数器值时, TAPWM 管脚上的脉冲保持高电平。一个脉冲周期等于 256 个  $t_{CLK}$  (见 [图 11-3](#))。

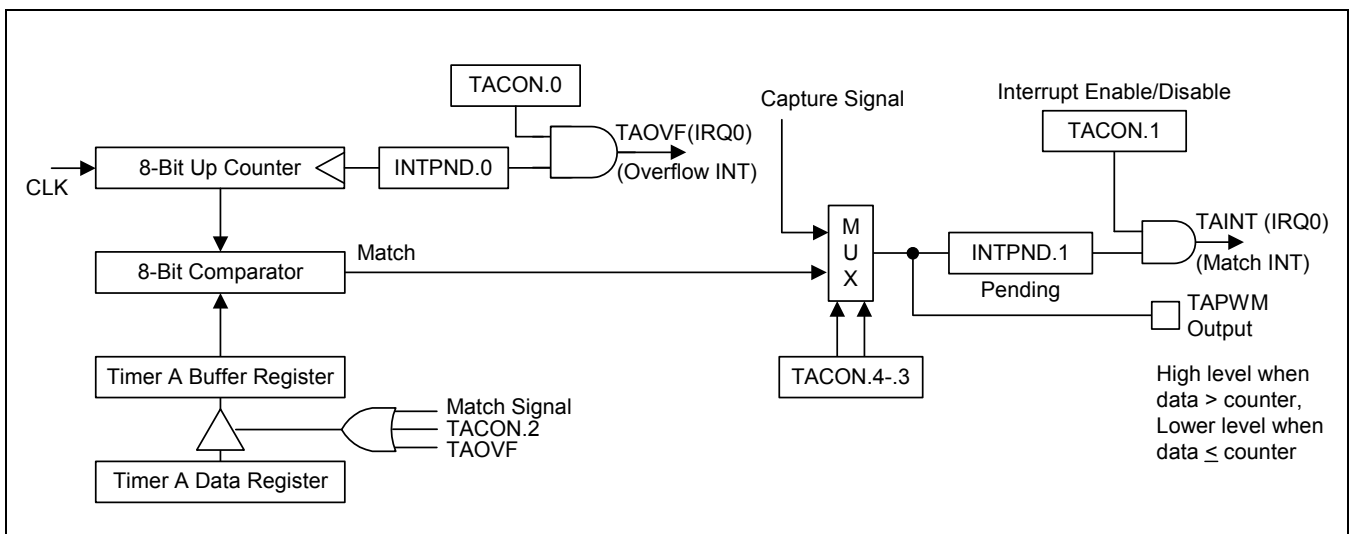


图 11-3 简化的 Timer A 功能框图: PWM 模式



### 11.1.5 CAPTURE（捕获）模式

在捕获模式中，当 TACAP 管脚上检测到一个信号沿时，把当前计数器的值载入 Timer A 数据寄存器中。可以选择上升沿或者下降沿触发。

Timer A 也提供了捕获输入源：TACAP 管脚上的信号沿。通过设置 P3 口低字节控制寄存器 P3CONL.7–6（地址：EBH，Set 1，Bank 1）的 Timer A 捕获输入选择位来选择捕获输入。当 P3CONL.7–6 位为“00B”时，选择作为 TACAP 输入。

Timer A 的两种中断均可用在捕获模式：当计数器溢出时产生 Timer A 溢出中断；而当计数器值载入到 Timer A 数据寄存器时产生 Timer A 匹配/捕获中断。

通过读取 TADATA 中的捕获数据值，并为 Timer A 假定一个特定的时钟频率，就可以计算出 TACAP 管脚上输入信号的脉冲宽度（持续时间），见 [图 11-4](#)。

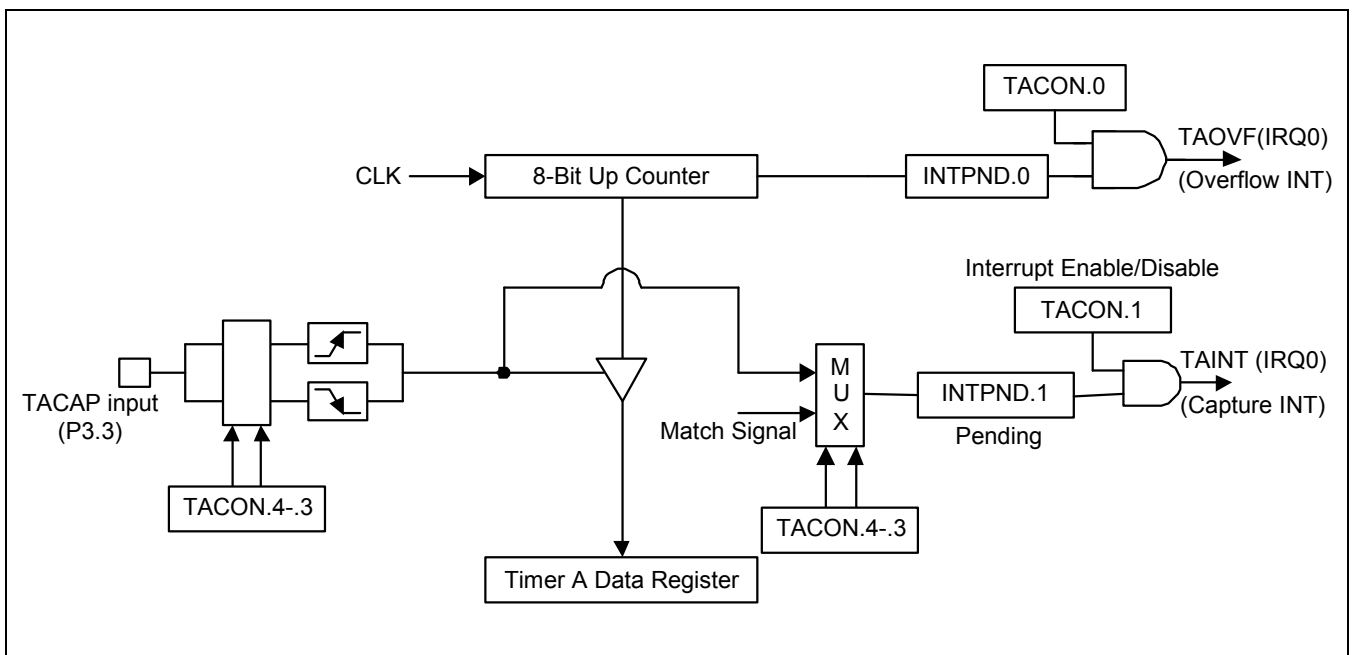


图 11-4 简化的 Timer A 功能框图：Capture（捕获）模式

11.1.6 模块框图

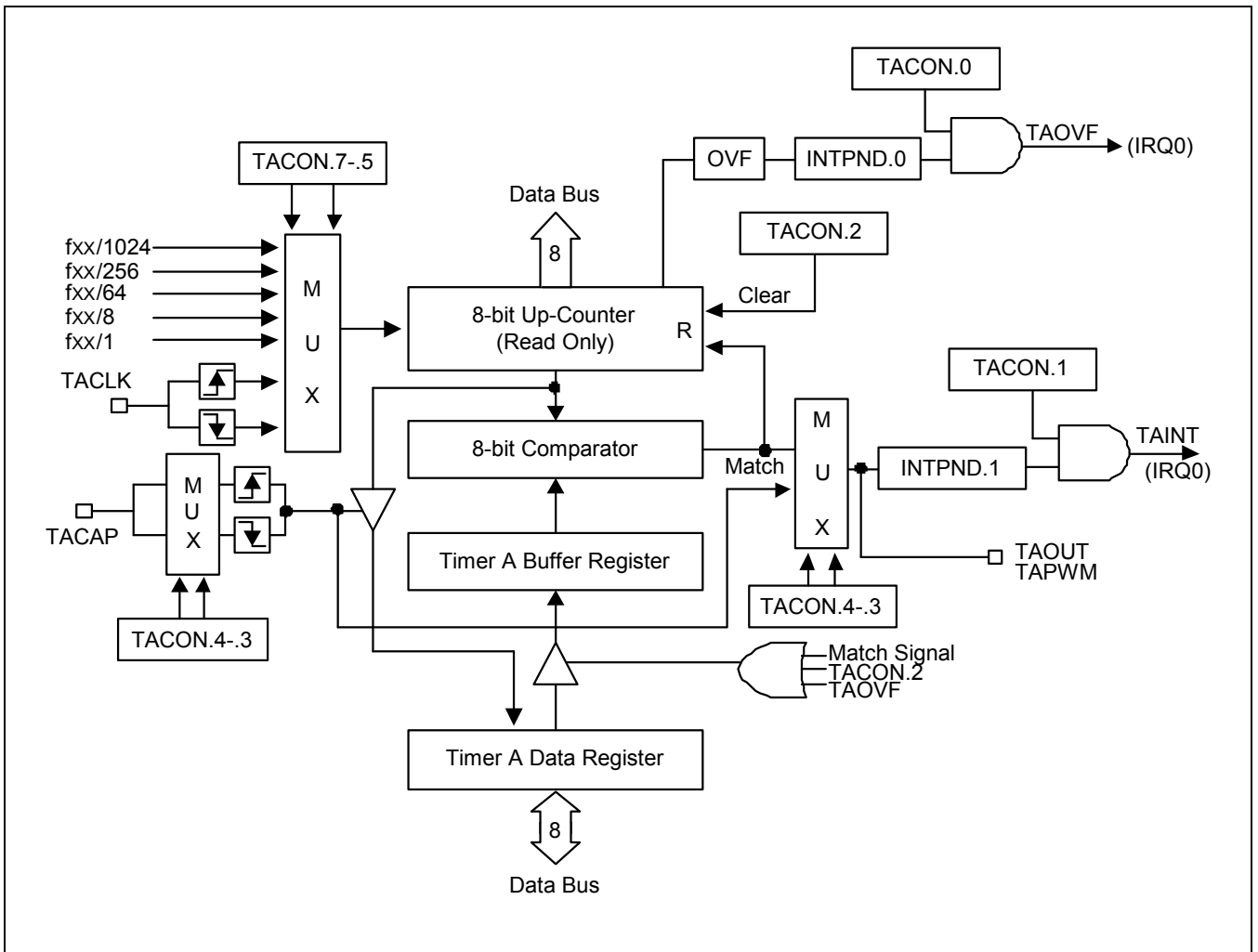


图 11-5 Timer A 功能模块框图

## 11.2 8 位 TIMER B

### 11.2.1 概述

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X MCU 有一个 8 位计数器叫做 Timer B, 用于产生遥控器信号的载波频率。

硬件会自动清除 Timer B 的中断标志位。

Timer B 有 2 个功能:

- 作为普通的定时时钟, 在编程的时间间隔内产生 Timer B 中断
- 给 8 位 Timer/Counter 模块 Timer B 提供时钟源, 用于产生 Timer B 溢出中断

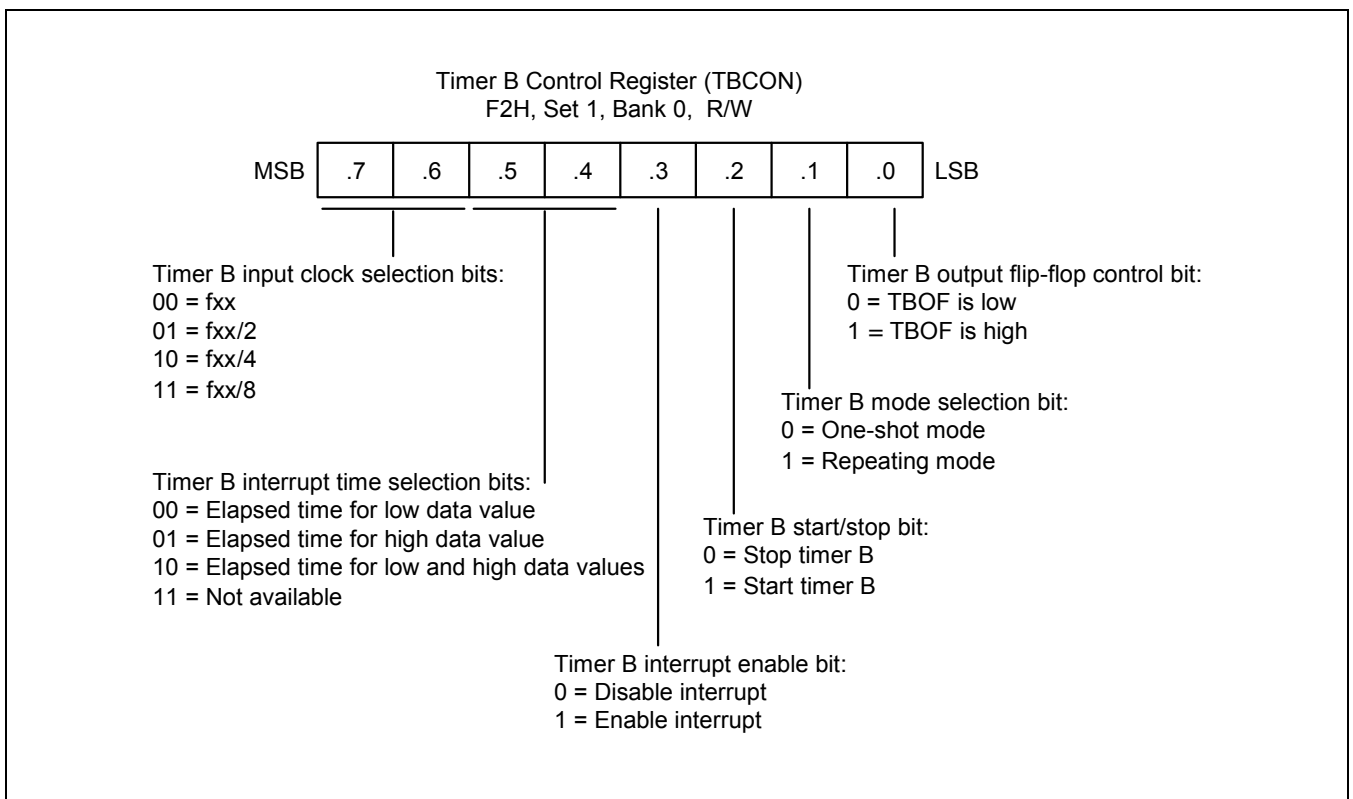


图 11-6 Timer B 控制寄存器

## 11.2.2 模块框图

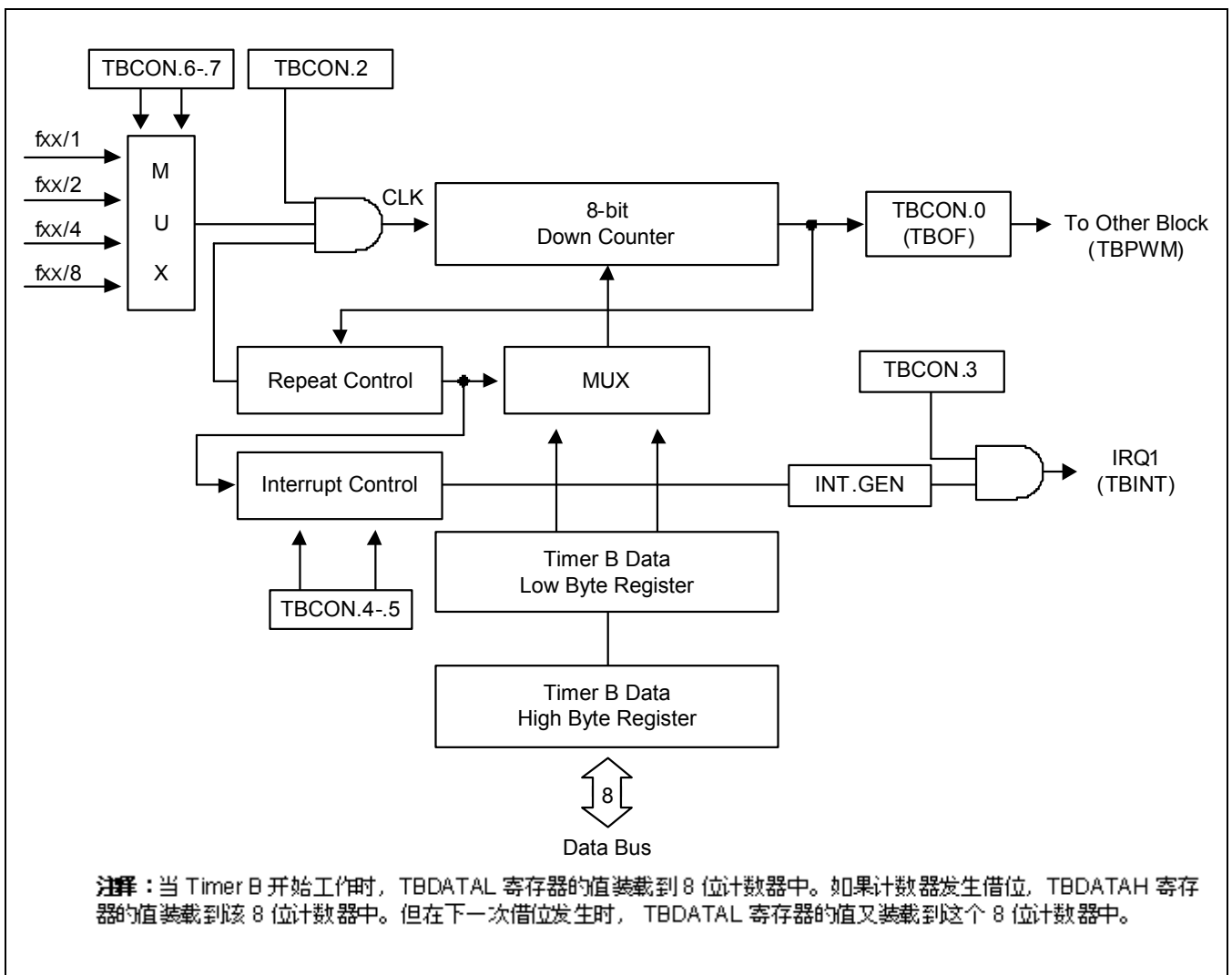
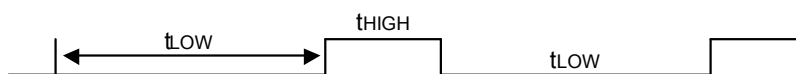


图 11-7 Timer B 功能模块框图

## 11.2.3 TIMER B 脉冲宽度计算



为产生如上所示的 Repeat（重复）波形，需要由低周期时间  $t_{LOW}$  和高周期时间  $t_{HIGH}$ 。

当 TBOF = 0 时，

$t_{LOW} = (TBDATAL + 2) \times 1/f_x$ ,  $0H < TBDATAL < 100H$ , 此时  $f_x =$  已选的时钟。

$t_{HIGH} = (TBDATAH + 2) \times 1/f_x$ ,  $0H < TBDATAH < 100H$ , 此时  $f_x =$  已选的时钟。

当 TBOF = 1 时，

$t_{LOW} = (TBDATAH + 2) \times 1/f_x$ ,  $0H < TBDATAH < 100H$ , 此时  $f_x =$  已选的时钟。

$t_{HIGH} = (TBDATAL + 2) \times 1/f_x$ ,  $0H < TBDATAL < 100H$ , 此时  $f_x =$  已选的时钟。

为了获得  $t_{LOW} = 24\mu s$  以及  $t_{HIGH} = 15\mu s$ 。  $f_{OSC} = 4MHz$ ,  $f_x = 4MHz/4 = 1MHz$

当 TBOF = 0 时，

$t_{LOW} = 24\mu s = (TBDATAL + 2) / f_x = (TBDATAL + 2) \times 1\mu s$ , TBDATAL = 22。

$t_{HIGH} = 15\mu s = (TBDATAH + 2) / f_x = (TBDATAH + 2) \times 1\mu s$ , TBDATAH = 13。

当 TBOF = 1 时，

$t_{HIGH} = 15\mu s = (TBDATAL + 2) / f_x = (TBDATAL + 2) \times 1\mu s$ , TBDATAL = 13。

$t_{LOW} = 24\mu s = (TBDATAH + 2) / f_x = (TBDATAH + 2) \times 1\mu s$ , TBDATAH = 22。

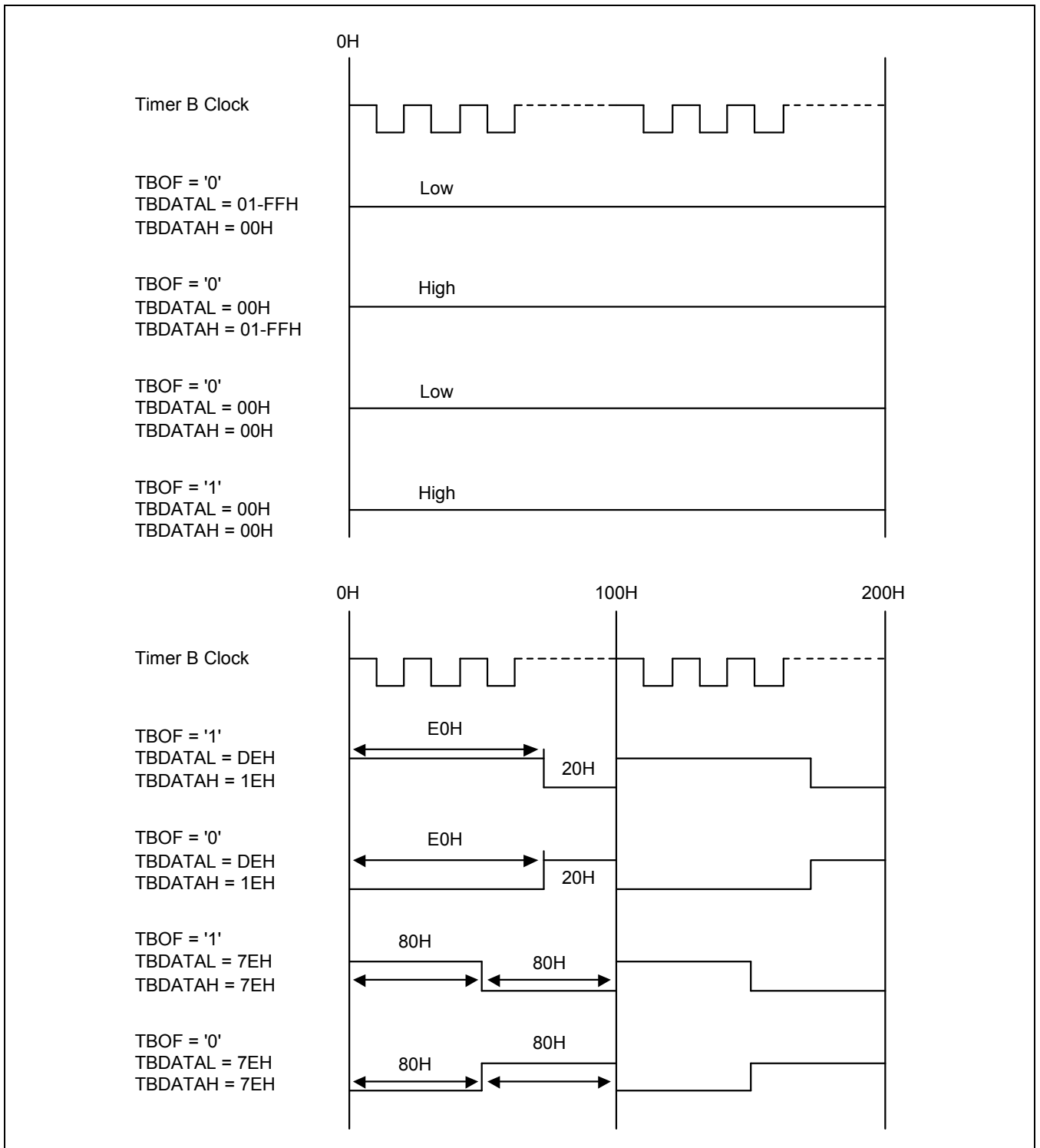
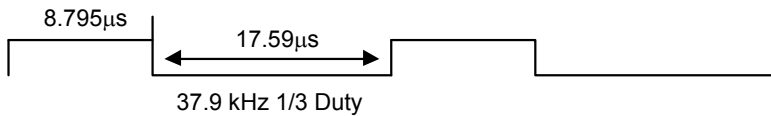


图 11-8 Timer B 在 Repeat (重复) 模式下输出的触发器波形

## 编程实例 11-1 如何在 P3.0 处产生 38kHz, 1/3 占空比信号

本例中设置 Timer B 为 Repeat (重复) 模式, 并且设置时钟频率为 Timer B 时钟源, 以及设置 TBDATAH 和 TBATAL 获得一个 38kHz, 1/3 占空比得载波频率。该编程参量如下:



- Timer B 用于 Repeat (重复) 模式
- 时钟频率为 4 MHz (0.25µs)
- $TBDATAH = 8.795\mu s / 0.25\mu s = 35.18$ ,  $TBATAL = 17.59\mu s / 0.25\mu s = 70.36$
- 设置 P3.0 为 TBPWM 模式

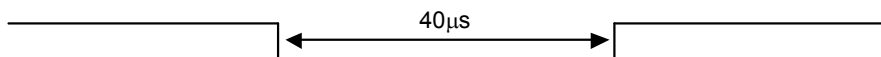
```

START:   ORG    0100H           ; 复位地址
         DI
         .
         .
         .
         LD    TBATAL, #(70-2) ; 设置 17.5µs
         LD    TBATAH, #(35-2) ; 设置 8.75µs
         LD    TBCON, #00000110B ; 时钟源 ← fxx
                                   ; 禁止 Timer B 中断
                                   ; 选择 Timer B 为 Repeat (重复) 模式
                                   ; 启动 Timer B 工作
                                   ; 设置 Timer B 输出触发 (TBOF) 高
         LD    P3CONL, #02H    ; 设置 P3.0 为 TBPWM 模式
                                   ; 该指令在 P3.0 处产生 38 kHz, 1/3 占空比脉冲信号
         .
         .
         .

```

## 编程实例 11-2 如何在 P3.0 处产生一个脉冲

本例中设置 Timer B 为 One-shot (单次) 模式, 并且设置时钟频率为 Timer B 时钟源, 以及设置 TBDATAH 和 TBATAL 以获得一个 40 $\mu$ s 宽度脉冲。该编程参量如下:



- Timer B 用于 One-shot (单次) 模式
- 时钟频率为 4 MHz (1 clock = 0.25 $\mu$ s)
- TBDATAH = 40 $\mu$ s / 0.25 $\mu$ s = 160, TBATAL = 1
- 设置 P3.0 为 TBPWM 模式

```

START:      ORG    0100H ;      复位地址
            DI
            .
            .
            LD    TBDATAH,# (160-2) ;      设置 40 $\mu$ s
            LD    TBATAL,# 1 ;      设置为除 00H 外的任何值
            LD    TBCON,#00000001B ;      时钟源  $\leftarrow$  fOSC
            ;      禁止 Timer B 中断
            ;      设置 Timer B 为 One-shot (单次) 模式
            ;      停止 Timer B 工作
            ;      设置 Timer B 输出触发 (TBOF) 高
            LD    P3CONL, #02H ;      设置 P3.0 为 TBPWM 模式
            .
            .
PULSE_OUT: LD    TBCON,#00000101B ;      启动 Timer B 工作
            ;      为在此刻获得该脉冲
            ;      该指令执行后, 在脉冲下降沿开始前需要 0.75 $\mu$ s.
            .
            .

```



# 12

## 16 位 TIMER 0/1

### 12.1 16 位TIMER 0

#### 12.1.1 概述

这个16位Timer 是一个16位的通用timer/counter。Timer 0 通过设置T0CON寄存器可以选择任定时工作模式。

Timer 0 有如下的功能模块：

- 带多路复用的时钟分频器 (fxx /256, fxx C/64, fxx C/8, fxx /1)
- TBOF提供其中一种时钟频率
- 1个16位计数器 ((T0CNTH/L), 1个16 位比较器以及1个16位数据寄存器 (T0DATAH/L))
- 产生Timer 0中断 (IRQ2, 中断向量地址 E2H)
- Timer 0 控制寄存器T0CON (set 1, bank 0, E3H, 读/写)

### 12.2 功能描述

#### 12.2.1 定时模式

Timer 0可产生一种中断，匹配中断(TOINT)。TOINT中断等级为IRQ2，分配了独立的向量地址E2H。

中断响应以后，TOINT标志位自动硬件清零。即使TOINT禁止，应用服务程序仍然可以通过软件检测到TOINT的标志位并运行它的子程序。当这种情况发生时，必须在应用子程序里给TOINT标志位写“0”来清零标志位。

在定时模式中，当计数器的值与Timer 0参考数据寄存器T0DATAH/T0DATAL的值相等时，生成一个匹配信号。匹配信号生成一个Timer 0匹配中断 (TOINT, 向量地址 E2H) 同时计数器清零。

例如，如果写0010H 到 T0DATAH/T0DATAL，T0CON写0FH，计数器计数到10H，产生T0中断请求，计数器值复位，开始重新计数。

## 12.2.2 TIMER 0 控制寄存器 (T0CON)

Timer 0 控制寄存器(T0CON)可以:

- 使能Timer 0 工作(定时模式)
- 选择Timer 0 输入时钟频率
- 清零Timer 0 计数器 T0CNT
- 使能Timer 0中断并清零中断标志位

T0CON位于set 1, Bank 0的地址E3H, 可以通过寄存器寻址模式读/写。

复位操作将置T0CON为“00H”, 这将设置Timer 0 禁止定时模式, 选择TBOF的输入的时钟频率, 并禁止所有Timer 0 中断。可以通过写“1”到T0CON.3随时清零Timer 0计数器。

要是能Timer 0中断(IRQ2, 向量 E2H), 必须写“1”到T0CON.1和T0CON.2。

为了产生精确的定时, 必须写0到T0CON.3来清零计数器和中断标志位。当T0INT禁止时, 要检测中断标志位, 应用程序可以轮询中断标志位T0CON.0。检测到标志位为“1”时, Timer 0匹配/捕获中断挂起。

当中断响应后, 可以通过软件对中断标志位T0CON.0写“0”来清除中断标志。

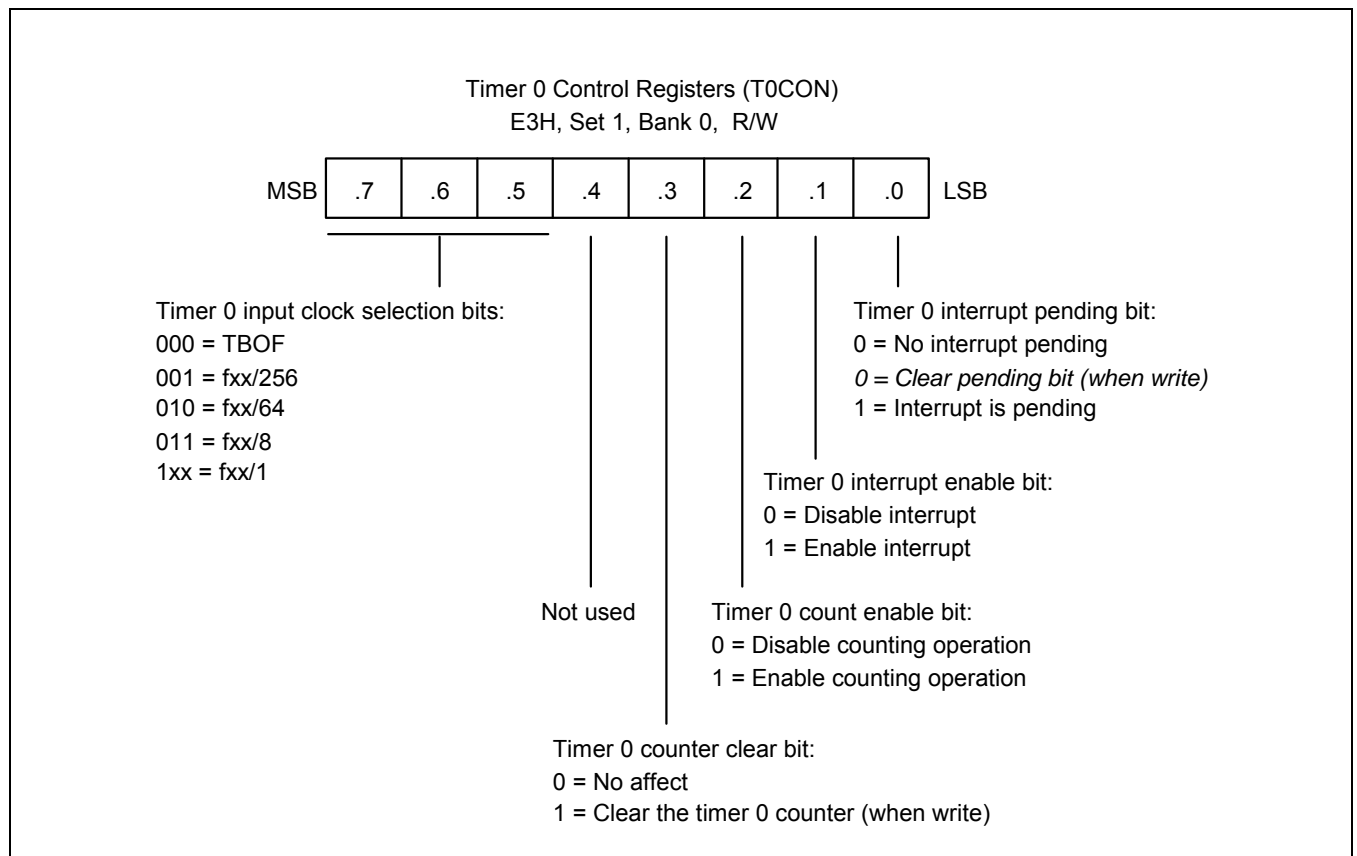


图 12-1 Timer 0 控制寄存器(T0CON)

12.2.3 框图

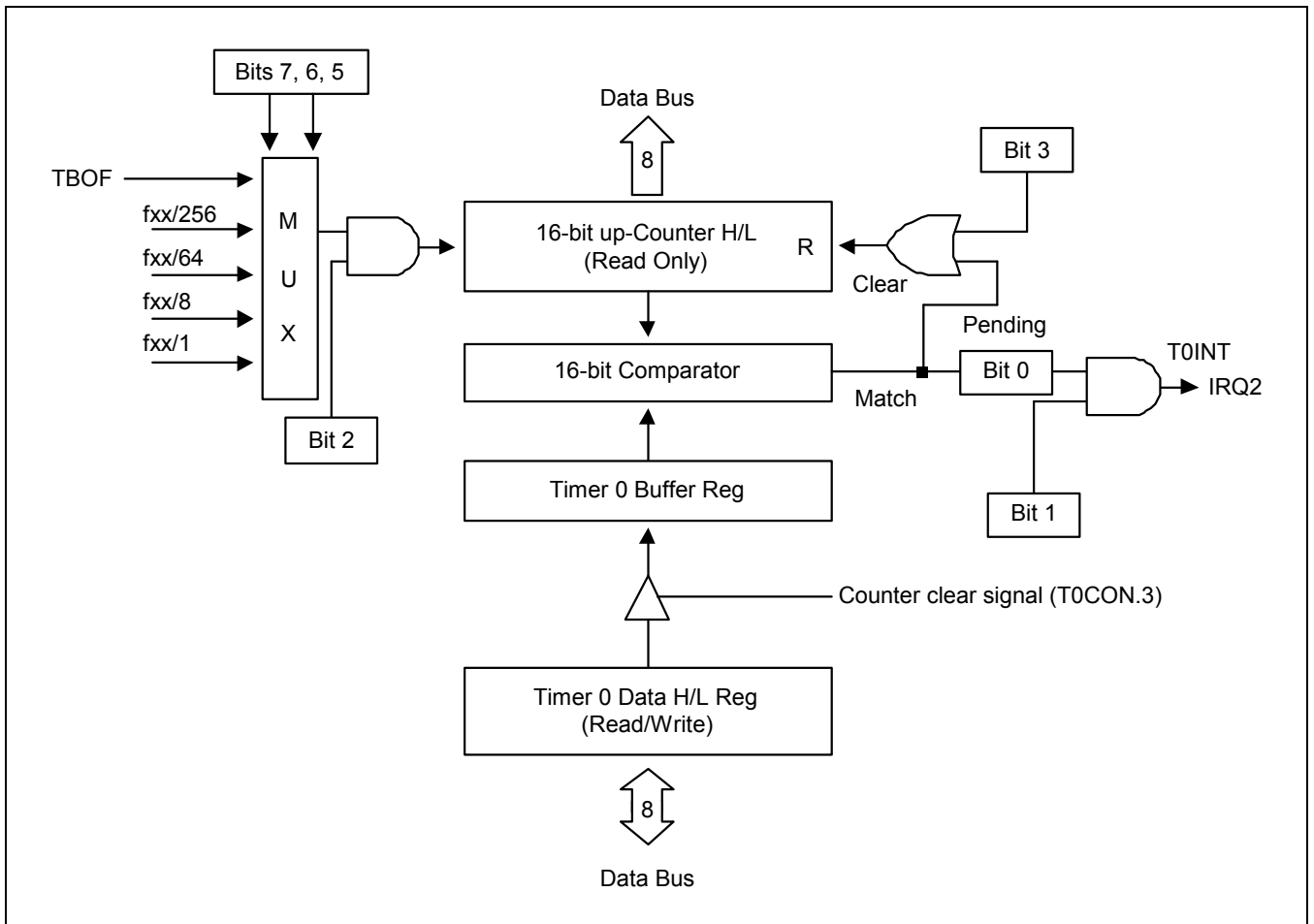


图 12-2 Timer 0 功能框图

## 12.3 16 位TIMER 1

### 12.3.1 概述

16位Timer 1是一个16位的通用timer。Timer 1 有3种工作模式，通过设置T1CON寄存器可以选择任一工作模式：

- 定时模式 (T1OUT 输出)
- 捕获输入模式 (T1CAP的上升沿或下降沿触发)
- PWM 模式 (T1PWM)

Timer 1 有如下的功能模块：

- 带多路复用的时钟分频器 (fxx /1024, fxx /256, fxx /64, fxx/8, fxx /1)
- 外部时钟输入管脚 (T1CLK)
- 1个16位计数器 (T1CNTH/L)，1个16 位比较器以及1个16位数据寄存器(T1DATAH/L)
- 捕获输入管脚 (T1CAP)，PWM或匹配输出管脚 (T1PWM, T1OUT)
- 产生Timer 1溢出中断 (IRQ3, 中断向量地址 E6H) 和匹配/捕获中断 (IRQ3, 中断向量地址 E4H)
- Timer 1控制寄存器 T1CON (set 1, Bank 0, EBH, 读/写)

### 12.3.2 TIMER 1 控制寄存器 (T1CON)

Timer 1 控制寄存器T1CON可以：

- 选择Timer 1工作模式 (定时模式, 捕获模式, PWM模式)
- 选择Timer 1 输入时钟频率
- 清零Timer 1 计数器 T1CNTH/T1CNTL
- 使能Timer 1 溢出中断或者使能Timer 1 匹配/捕获中断

T1CON 位于set 1, Bank 0的地址EBH, 可以通过寄存器寻址模式读/写。

复位操作将置T1CON为“00H”, 这将设置Timer 1 为定时模式, 选择输入的时钟频率为 $f_{xx}/1024$ , 并禁止所有Timer 1 中断。可以通过设置T1CON .7 - .5 为“111B”来禁止计数器操作。可以通过写“1”到T1CON.3随时清I零Timer 1计数器。

Timer 1溢出中断 (T1OVF) 的中断优先级为IRQ3, 中断向量地址为E6H。当Timer 1溢出中断发生时, 响应中断(IRQ3, 向量 E6H), 必须写“1”到T1CON.1. 要产生精确的定时, 应该同 过CPU写T1CON, 标志位由硬件自动清零。

要是能Timer 1 匹配/捕获中断, 必须写“1”到T1CON.1. 当T1INT或者T1OVF禁止时, 要检测匹配/捕获或者溢出中断标志位, 应用程序可以轮询中断标志位, 检测到标志位为“1”时, Timer 1匹配/捕获中断或者溢出中断挂起。当中断响应后, 必须通过软件对中断标志位写“0”来清除中断标志。

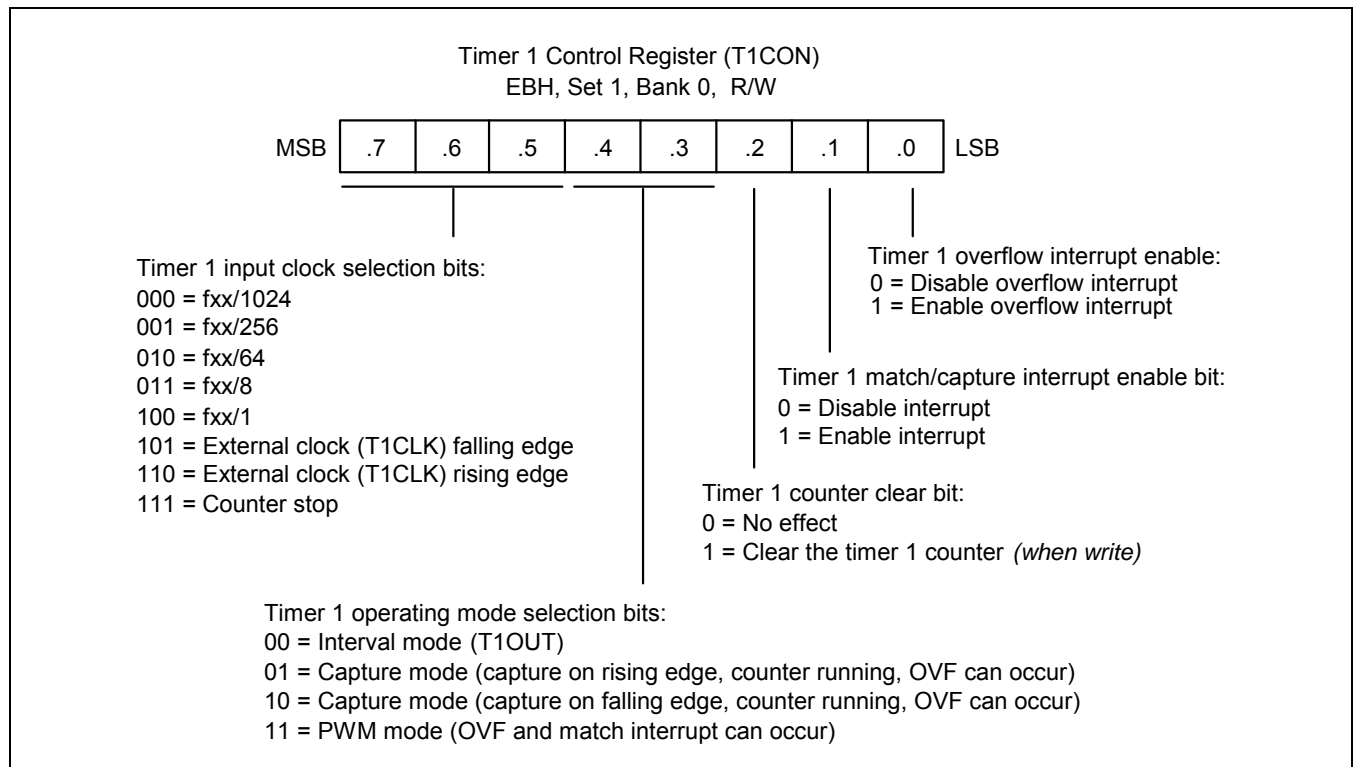


图 12-3 Timer 1 控制寄存器 (T1CON)

### 12.3.3 TIMER 1 功能描述

#### 12.3.3.1 Timer 1 中断 (IRQ2, 中断向量地址 E4H 和 E6H)

Timer 1 模块能产生2个中断, Timer 1 溢出中断 (T1OVF) 以及Timer 1 匹配/捕获中断 (T1INT)。  
T1OVF的中断优先级为IRQ3, 向量地址为E6H。T1INT的中断优先级也为IRQ3, 不同的是向量地址为E4H。

Timer 1 溢出中断标志在中断响应后由硬件自动清除或者在应用程序中写“0”到 INTPND.2软件清零。Timer 1 匹配/捕获中断必须通过在应用程序中写“0”到 INTPND.3软件清零。

#### 12.3.4 定时模式

在定时模式中, 当计数器的值与Timer 1参考数据寄存器T1DATAH/TD0DATAL的值相等时, 生成一个匹配信号。匹配信号生成一个Timer 1匹配中断 (T1INT, 向量地址 E4H) 同时计数器清零。

例如, 如果写“1087H”到 T1DATAH/T1DATAL, 计数器计数到“1087H”, 产生timer 1中断请求, 计数器值复位, 开始重新计数。每次匹配, timer 1输出的信号发生反转(见 [图 12-4](#))。

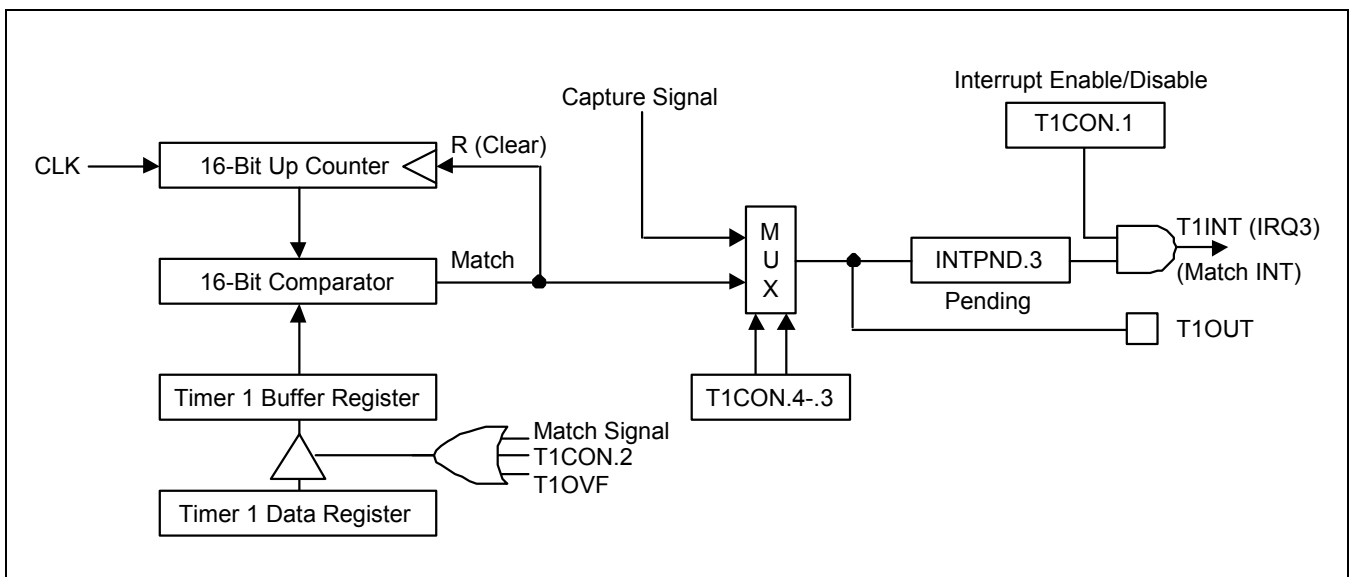


图 12-4 Timer 1 功能简图: 定时模式

### 12.3.5 PWM 模式

脉宽调制模块 (PWM)，用户可以控制管脚T1PWM输出的脉冲宽度 (持续时间)。

在在PWM模式下，当计数器的值与Timer 1 数据寄存器中的值相等时，产生一个匹配信号，但不清零计数器，它会连续工作，直到FFFFH时溢出，然后从0000H继续向上计数。

虽然在PWM模式下可以使用匹配或者溢出中断，但是这些中断在PWM应用中并不普遍。常用到的功能是，管脚T1 PWM 上的脉冲，在参考数据寄存器中的值小于等于计数器的值时保持低电平；而在参考数据寄存器中的值大于计数器的值时保持高电平。一个脉冲宽度等于 $t_{CLK} \cdot 65536$  (见[图 12-5](#))。

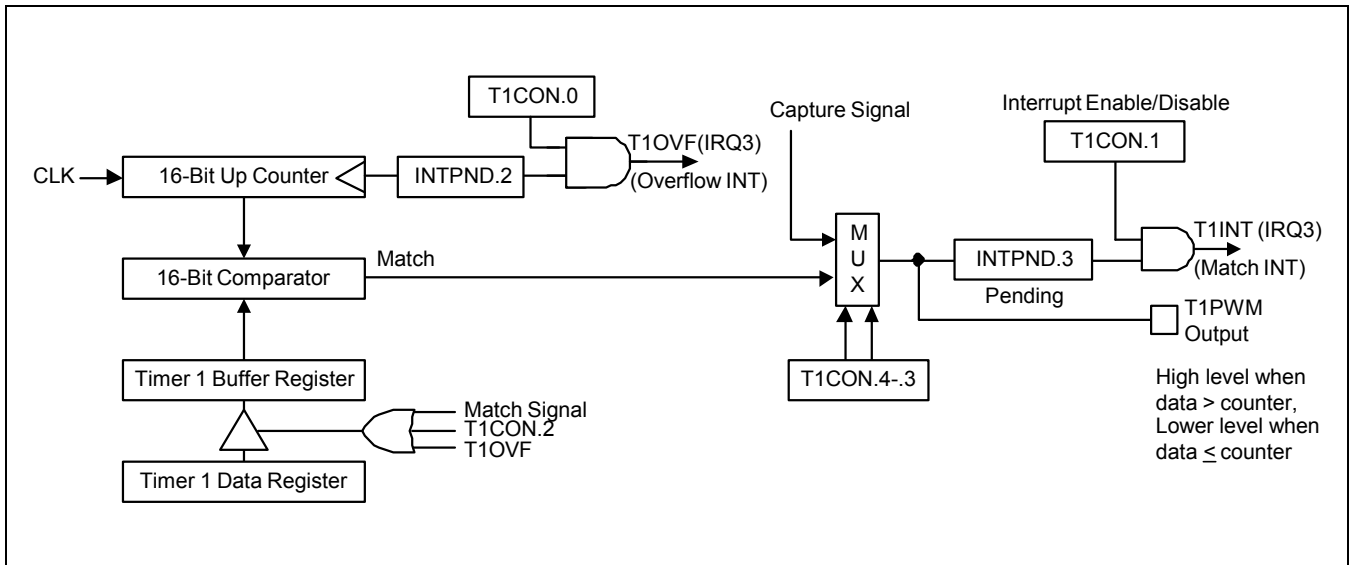


图 12-5 Timer 1 功能简图: PWM 模式

### 12.3.6 捕获模式

Timer 1 的捕获模式下，一旦管脚T1CAP上检测到信号边沿就将当前计数器的值载入Timer 1数据寄存器。用户可以选择上升沿或者下降沿来触发此操作。

Timer 1捕获输入源：管脚T1CAP上的信号边沿。可以通过设置P3高字节控制寄存器P1CONL.1–0, (set 1 bank0, E6H) 来选择捕获输入。当P1CONL.1–0,为"00"时，选择T1CAP输入。

Timer 1 的两种中断 均可用于捕获模式，当计数器溢出时产生Timer 1溢出中断；当计数器值载入Timer 1数据寄存器时产生Timer 1捕获中断。

通过读取T1DATAH/T1DATAL,寄存器中的值，并为Timer 1假设一个特定的时钟频率，就可以计算出管脚T1CAP上输入信号的脉冲宽度（持续时间）(见 [图 12-6](#))。

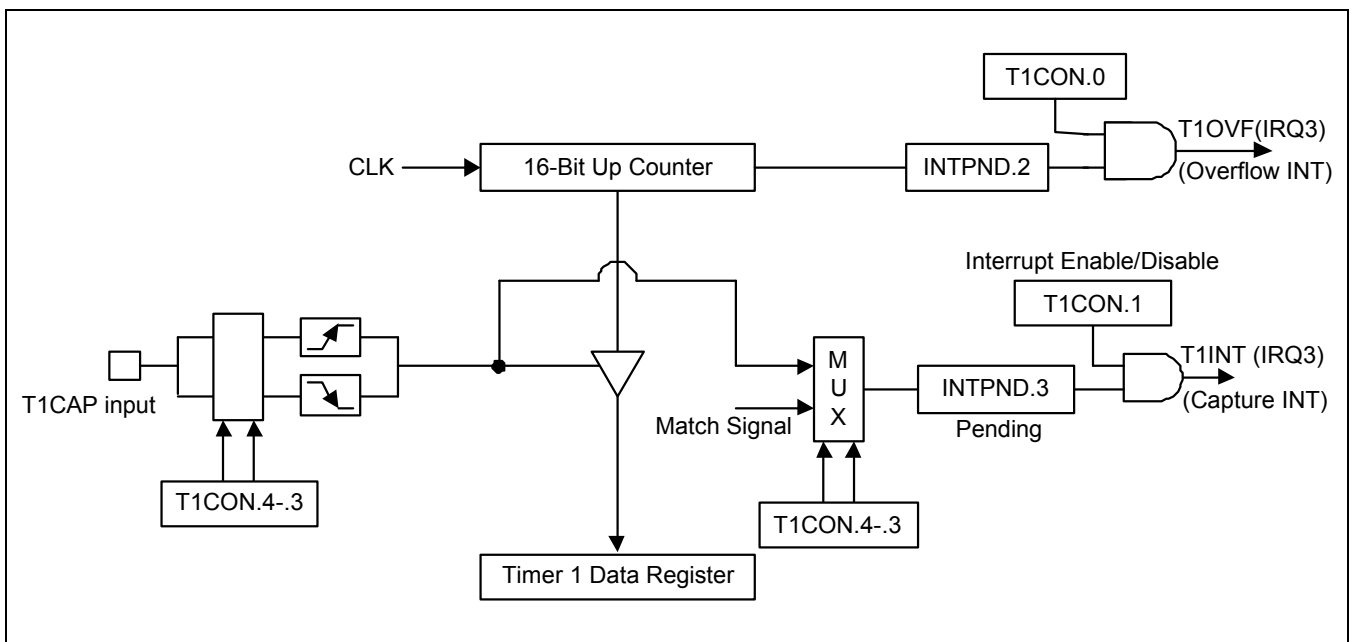


图 12-6 Timer 1 功能简图: 捕获模式



12.3.7 框图

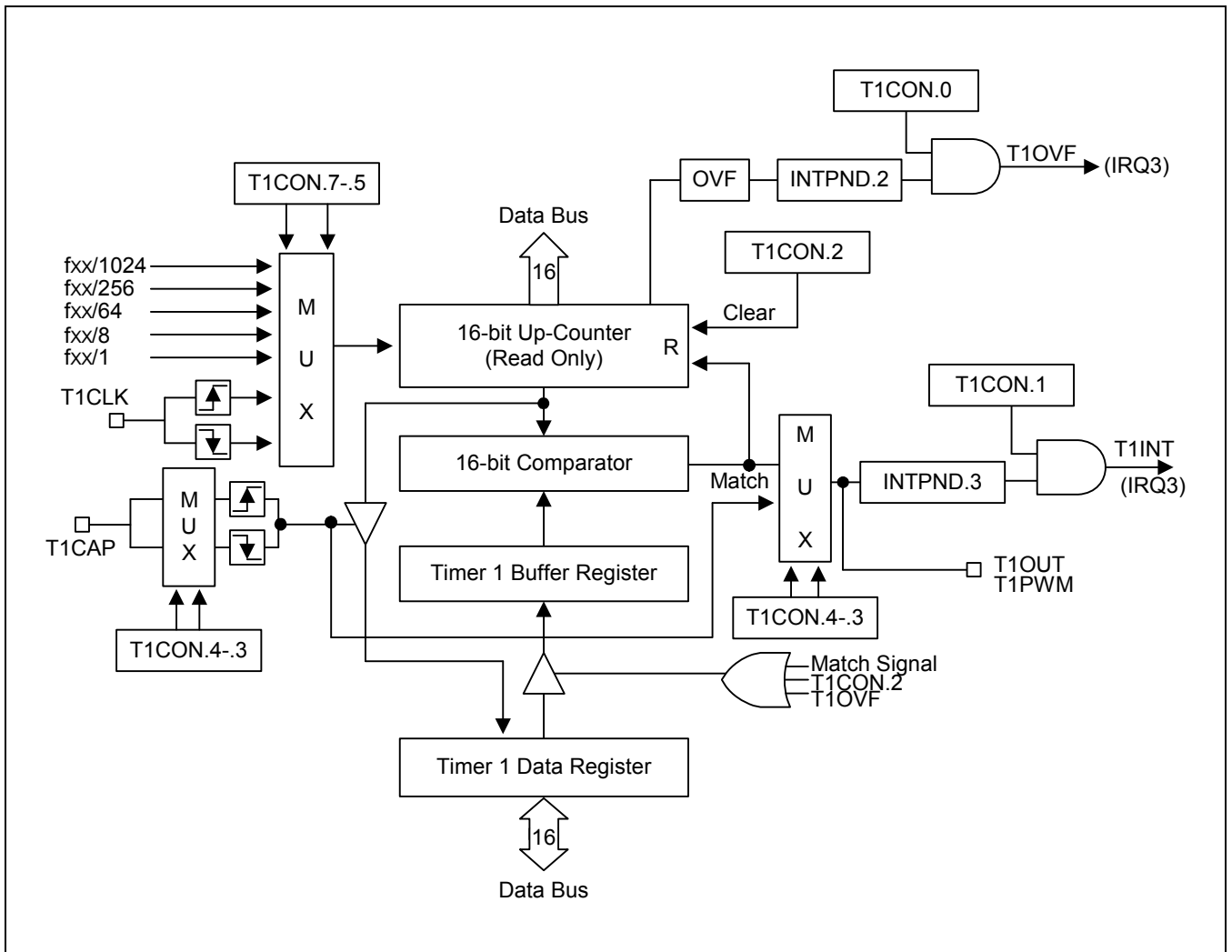


图 12-7 Timer 1 功能框图

# 13

## 钟表定时器

### 13.1 概述

钟表定时器的功能包括实时时钟，时间测量以及为系统时钟提供间隔定时。为了启动钟表定时器，将控制寄存器的第 1 位 (WTCON.1) 设置为 "1"。如果希望使用钟表定时器的溢出中断 (IRQ5, 地址 EEH)，还要将 WTCON.1 设置为 "1"。

钟表定时器溢出中断标志 (WTCON.0) 必须在应用程序中进行软件清零，也就是向 WTCON.0 写入 "0"。

钟表定时器启动之后经过一段时间，中断标志会被自动置 "1"，中断请求以 3.91ms, 0.25s, 0.5s 或 1s 的间隔发生，这由钟表定时器的速度设置 (WTCON.3-2) 决定。

钟表定时器可以产生稳定的 0.5 kHz, 1 kHz, 2 kHz 或 4 kHz 信号，送到蜂鸣器输出管脚 (BUZ)。将 WTCON.3 和 WTCON.2 设置为 "11b"，钟表定时器将工作于高速模式，每隔 1.955ms 产生一次中断。在程序调试时可使用高速模式来对事件定时。

钟表定时器为 LCD 控制器提供时钟频率 ( $f_{LCD}$ )。因此，如果禁止 watch timer，LCD 控制器将不能工作。

钟表定时器包含以下组成部分：

- 实时时钟和时间测量
- 时钟源可使用主系统时钟或副系统时钟
- LCD 控制器时钟 ( $f_{LCD}$ ) 生成
- 蜂鸣器频率输出引脚 (BUZ)
- 高速模式下的时序测试
- 钟表定时器溢出中断 (IRQ5, 地址 EEH) 发生
- 钟表定时器控制寄存器, WTCON (set 1, bank 0, D1H, 读/写)

### 13.1.1 钟表定时器控制寄存器 (WTCN: R/W)

钟表定时器控制寄存器，WTCN，用于选择定时器中断间隔和蜂鸣器信号、使能或禁止watch timer功能。

它位于set 1, bank 0, 地址D1H, 可通过寄存器寻址模式进行读/写。

复位后WTCN 清零, 这将禁用钟表定时器功能。

所以, 如果想使用钟表定时器, 必须对WTCN寄存器写入适当的值。

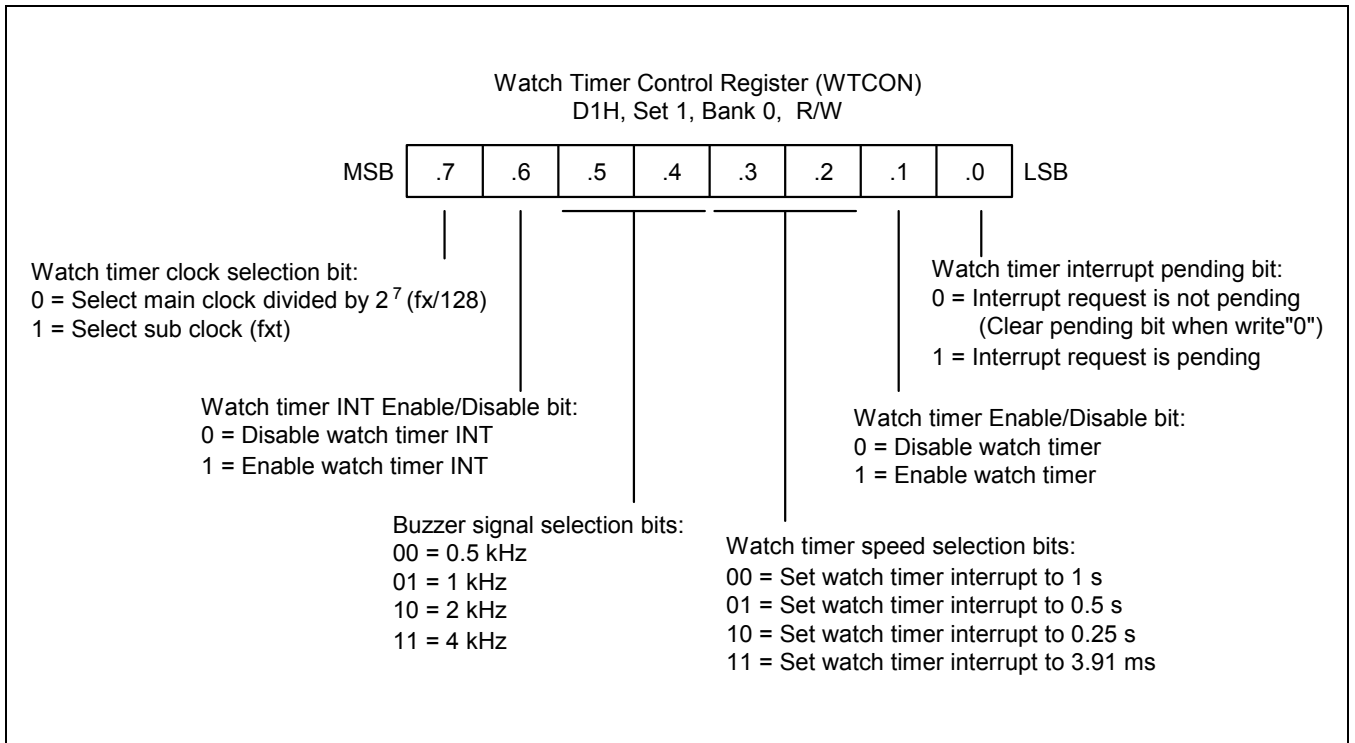


图 13-1 钟表定时器控制寄存器(WTCN)

13.1.2 钟表定时器电路图

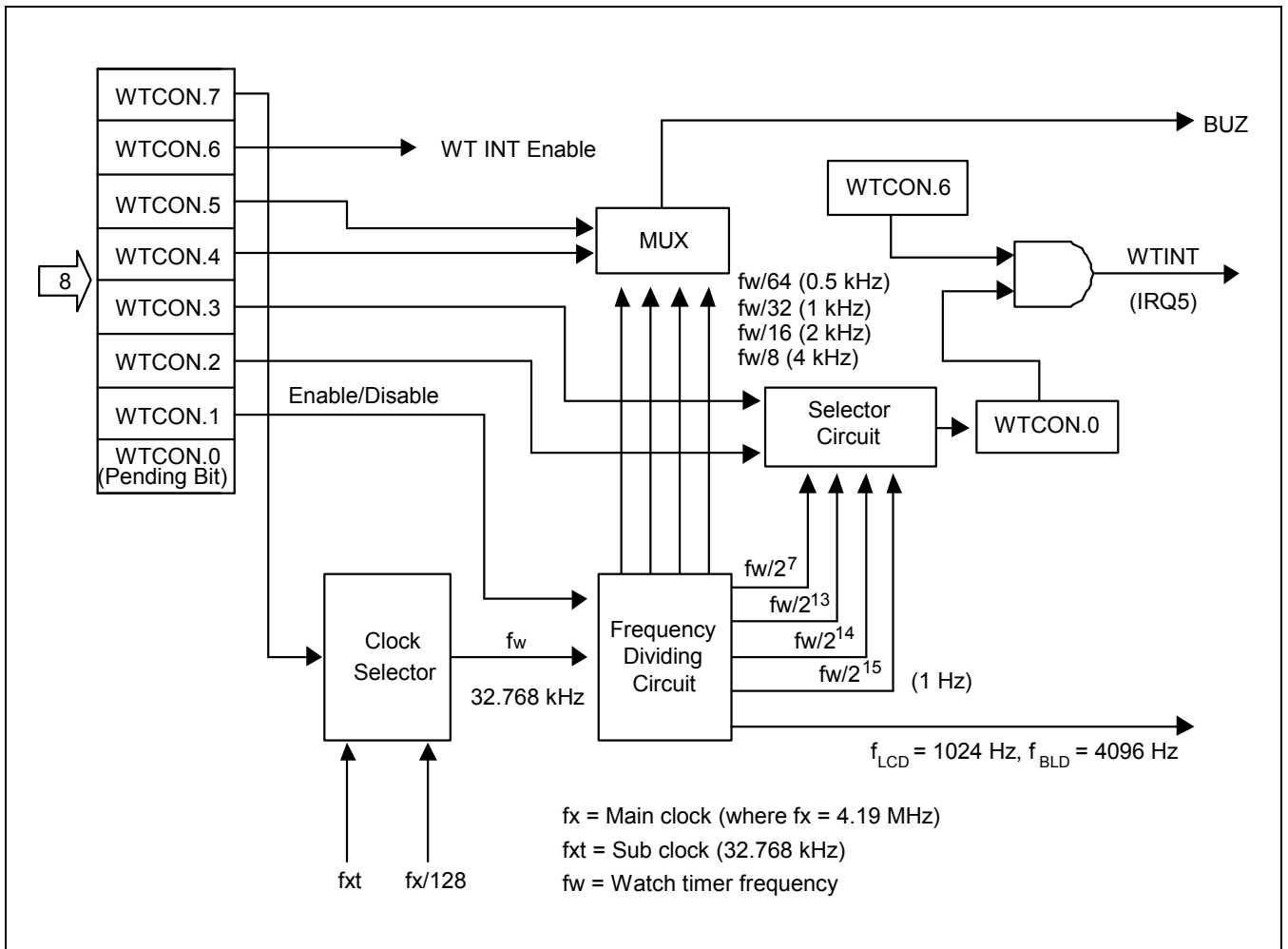


图 13-2 钟表定时器电路框图

# 14 LCD 控制器/驱动器

## 14.1 概述

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 微控制器可以直接驱动最多256点阵 (32 segments × 8 commons)的LCD面板。LCD模块包括以下部分：

- LCD 控制器/驱动器
- 存储显示数据的显示RAM
- 6 个common/segment 输出管脚(COM2/SEG0–COM7/SEG5)
- 32 个segment 输出管脚 (SEG6–SEG37)
- 2 个common 输出管脚 (COM0–COM1)
- 4 个LCD供电电压管脚( $V_{LC0}$ – $V_{LC3}$ )
- LCD内部/外部电阻偏压

LCD控制寄存器LCON用来打开/关闭LCD显示，选择帧频率，控制流向用于LCD显示分压电阻的电流。LCD对比度控制寄存器用来选择V<sub>LC</sub>电压。写到LCD显示RAM里的数据自动传输到segment信号管脚，不需要程序控制。

当选择副振作为LCD时钟源时，即使在主时钟STOP或者IDLE模式下LCD仍可以显示。

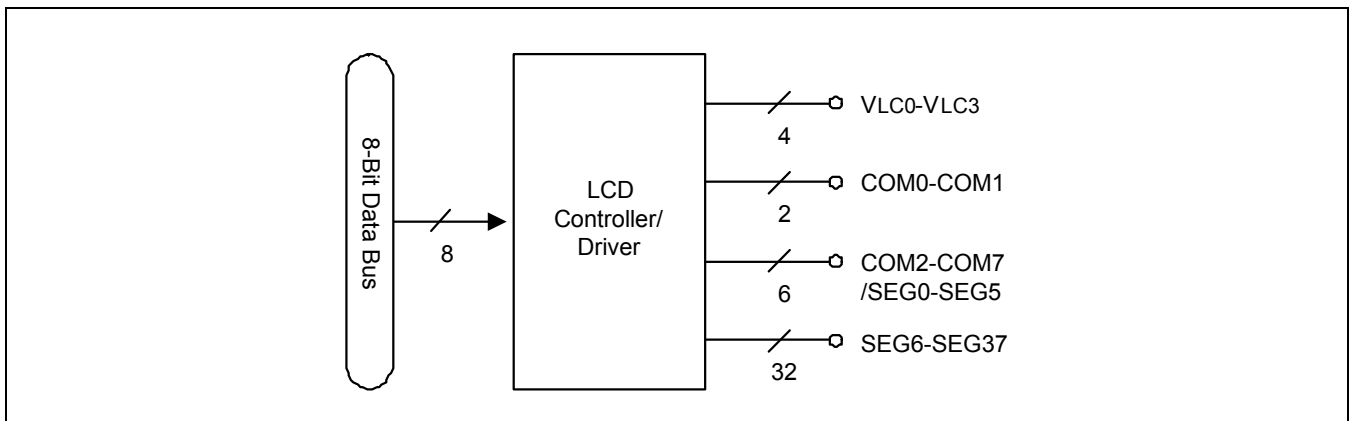


图 14-1 LCD功能框图

14.1.1 LCD 电路框图

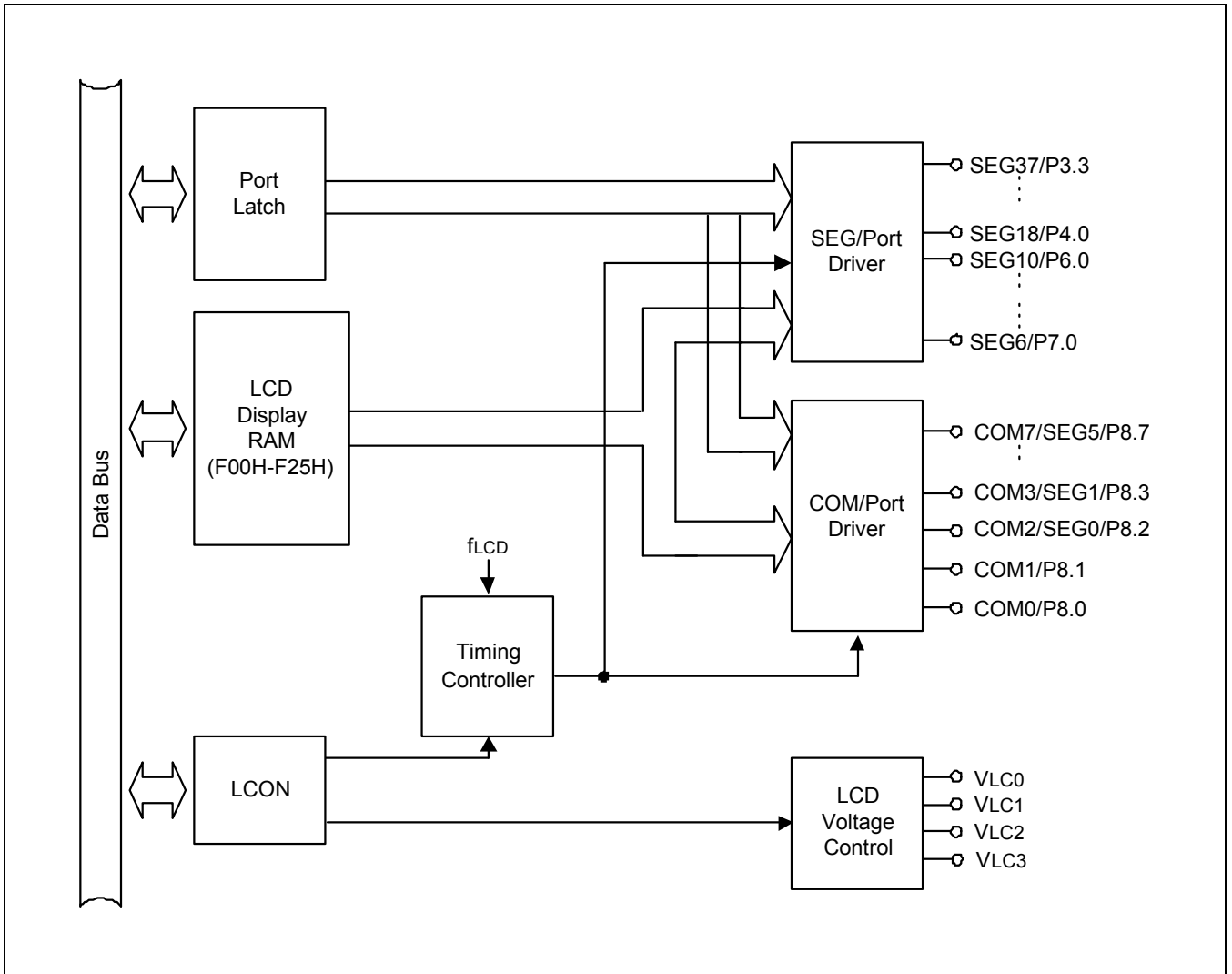


图 14-2 LCD 电路框图

### 14.1.2 LCD RAM 地址空间

Page 15中的RAM空间被用作LCD数据存储。 这些地址可由1位或者8位指令来寻址。  
当某个显示段的值为"1"时，LCD显示打开；当某个显示段的值为"0"时，LCD显示关闭。

显示RAM区的数据采用与信号fLCD 同步的直接内存访问（DMA）方式，通过管脚SEG0-SEG37送出。  
如果这些RAM地址没有用作LCD显示，也可以用作一般用途。

COM	Bit	SEG0	SEG1	SEG2	SEG3	SEG4	-----	SEG36	SEG37
COM0	.0								
COM1	.1								
COM2	.2								
COM3	.3	F00H	F01H	F02H	F03H	F04H	-----	F24H	F25H
COM4	.4								
COM5	.5								
COM6	.6								
COM7	.7								

图 14-3 LCD 显示数据 RAM 结构图

### 14.1.3 LCD 控制寄存器 (LCON)

LCON 位于 page 15, set 1, bank 1, D0H, 可读写, 寄存器寻址模式。它包含以下功能:

- LCD 占空比、偏压比选择
- LCD 时钟选择
- LCD 显示控制
- 内部/外部分压电阻偏压选择

LCON 寄存器可以控制打开/关闭 LCD 显示, 可以选择占空比和偏压比, 可以选择 LCD 时钟和控制 LCD 分压电阻的电流流向。复位后, LCON 的值被清零。LCD 显示关闭, 占空比选择 1/8, 偏压比选择 1/4, LCD 时钟选择 128Hz 并使能 LCD 内部分压电阻。

The LCD 时钟频率决定了每个 segment 输出 COM 信号的扫描频率。也被认为是 LCD 的帧频。既然 LCD 时钟由 watch timer(fw) 产生, LCD 显示打开的时候必须使能 watch timer。

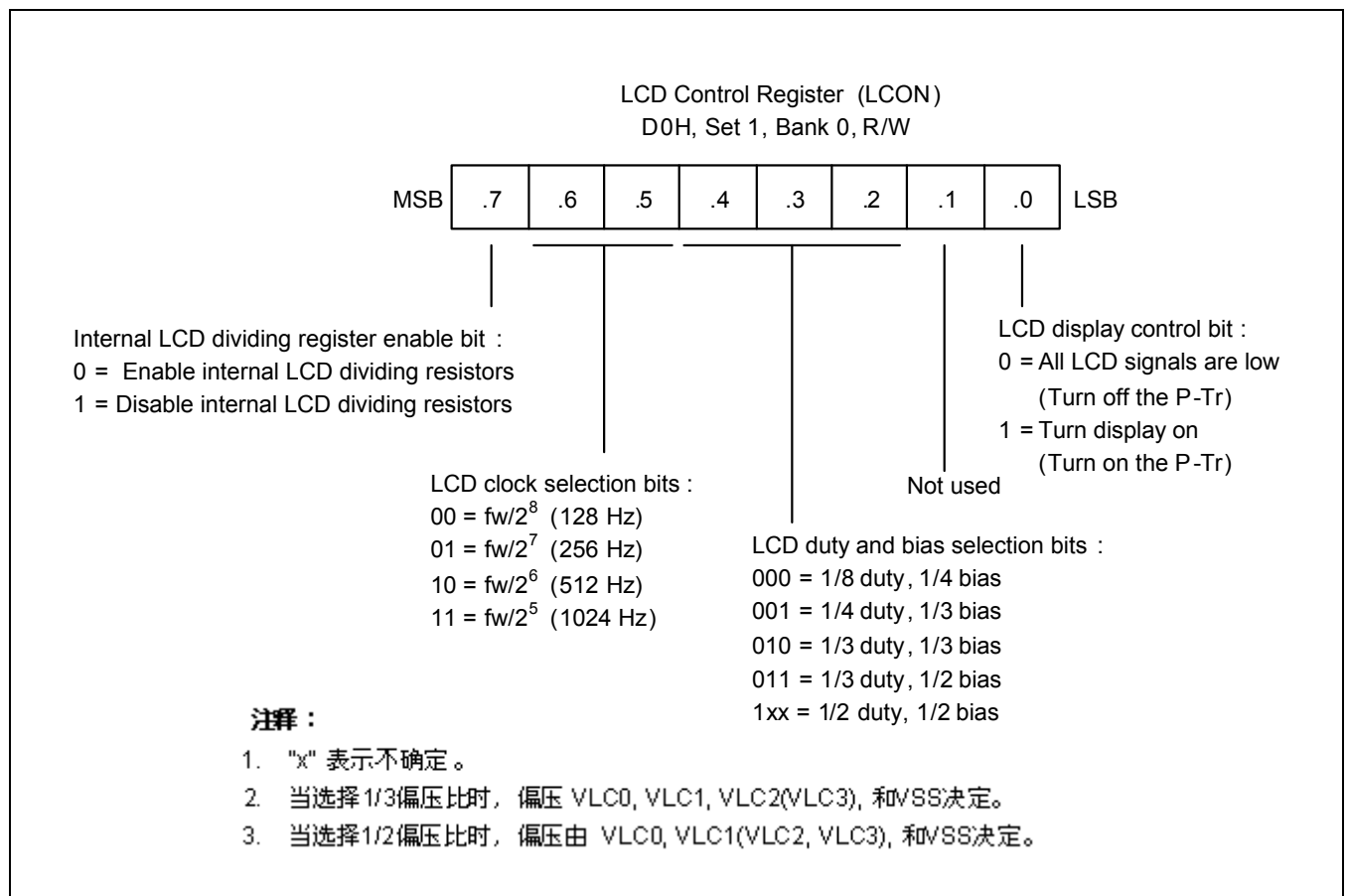


图 14-4 LCD 控制寄存器 (LCON)



14.1.4 LCD 分压电阻

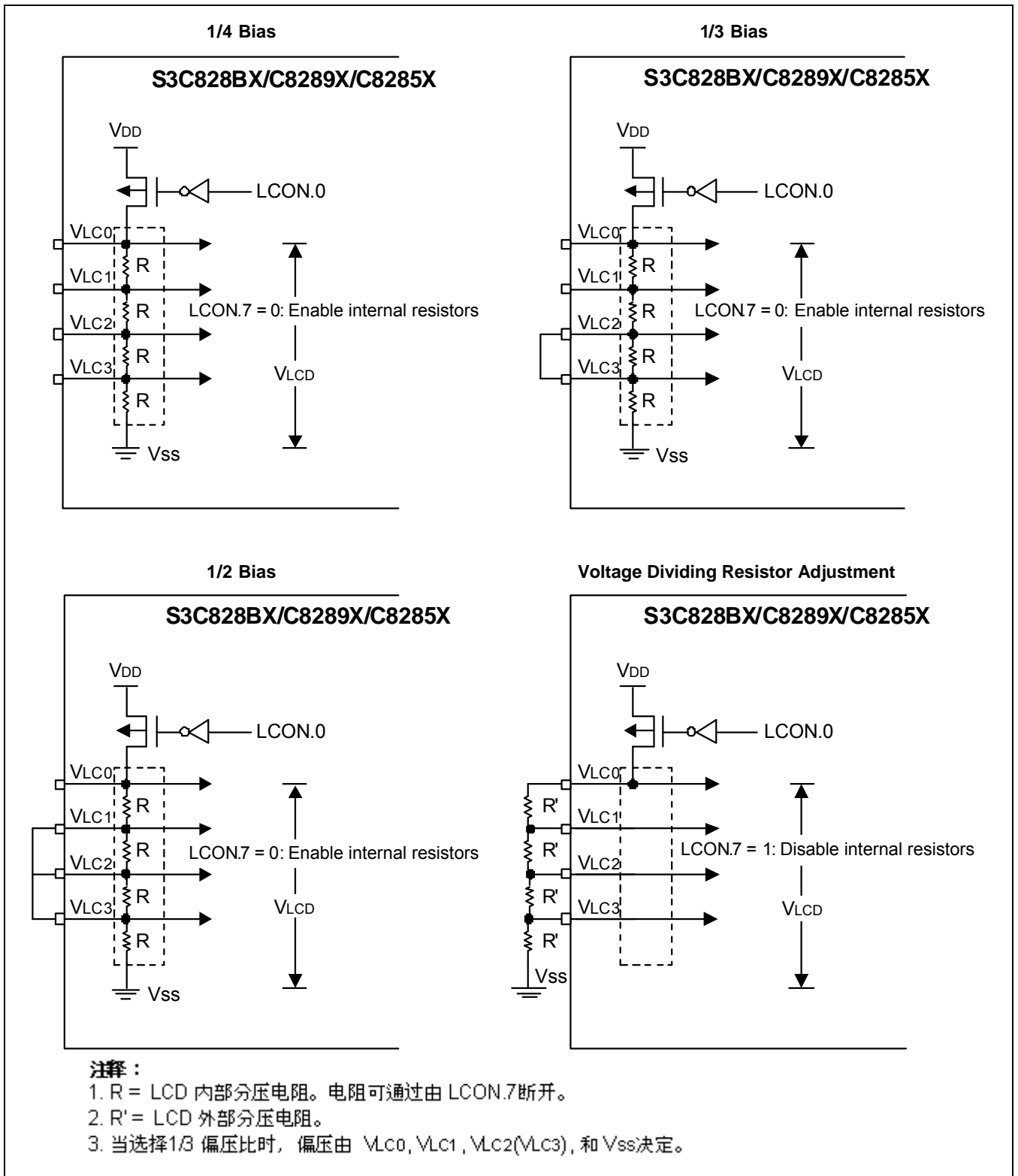


图 14-5 LCD 分压电阻连接

### 14.1.5 COMMON (COM) 信号

根据占空比选择不同的COM信号。

- 1/8 占空比，选择 COM0-COM7 (SEG6-SEG37)
- 1/4 占空比，选择 COM0-COM3 (SEG2-SEG37)
- 1/3 占空比，选择 COM0-COM2 (SEG1-SEG37)
- 1/2 占空比，选择 COM0-COM1 (SEG0-SEG37)

### 14.1.6 SEGMENT (SEG) 信号

38个LCD segment 信号管脚连到page 15的相应显示RAM区。RAM区的显示位与COM信号输出管脚同步。

当RAM区的显示位为 "1"时，相应的segment被选中。 当显示位为"0"时，说明相应的segment没有被选中。

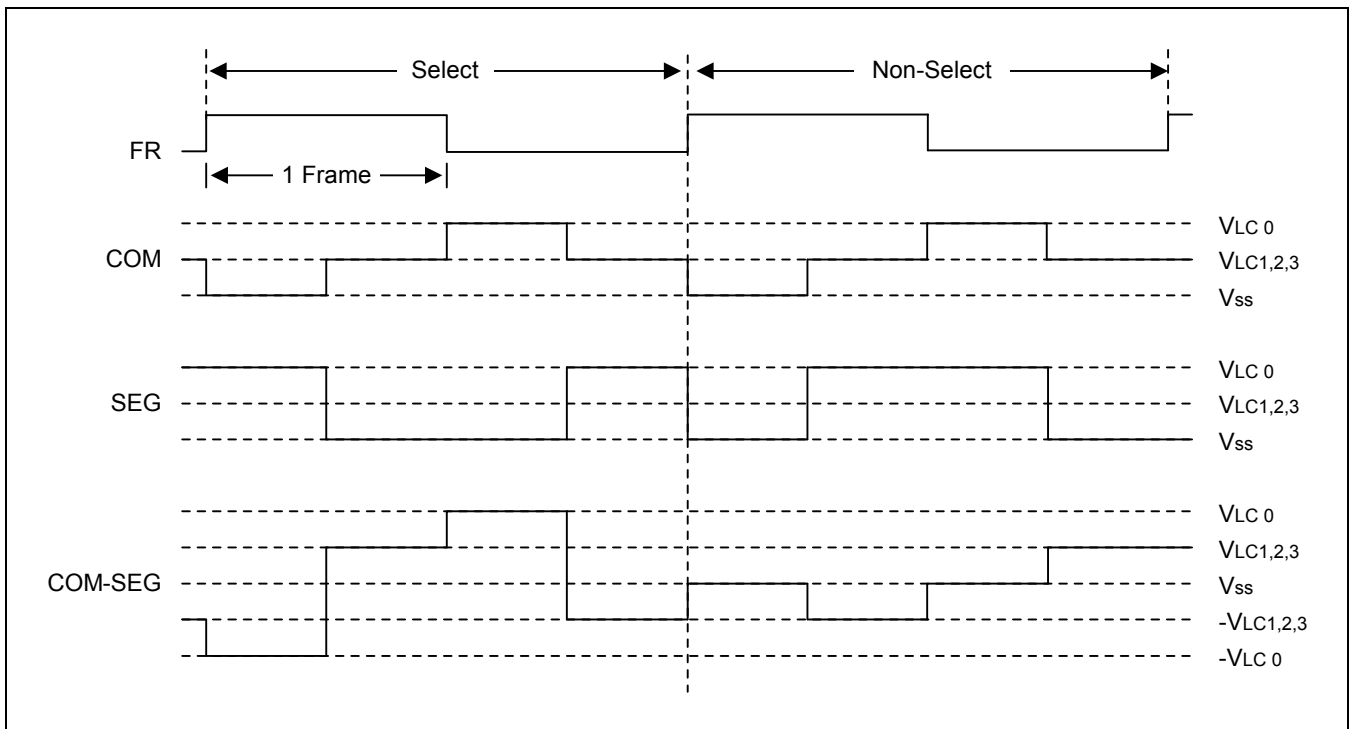


图 14-6 1/2占空比, 1/2偏压比显示模式下选中/非选中信号

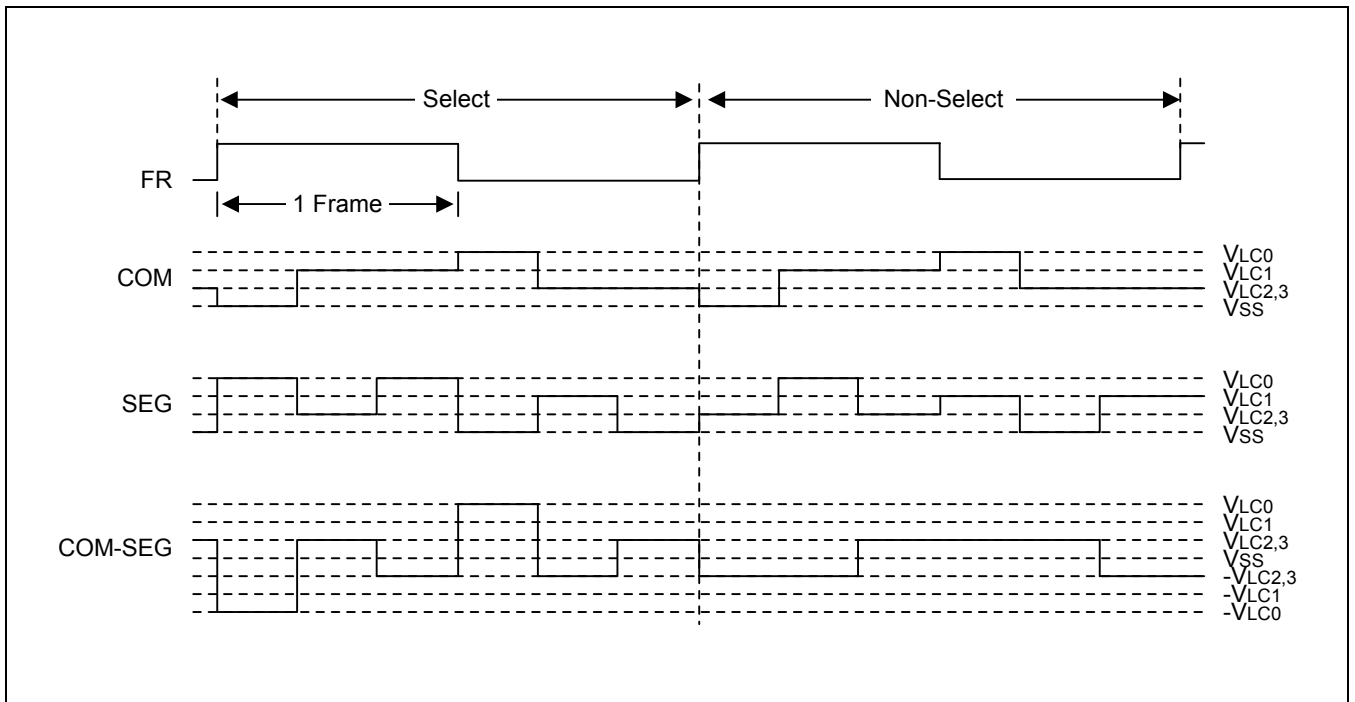


图 14-7 1/3占空比, 1/3偏压比显示模式下选中/非选中信号

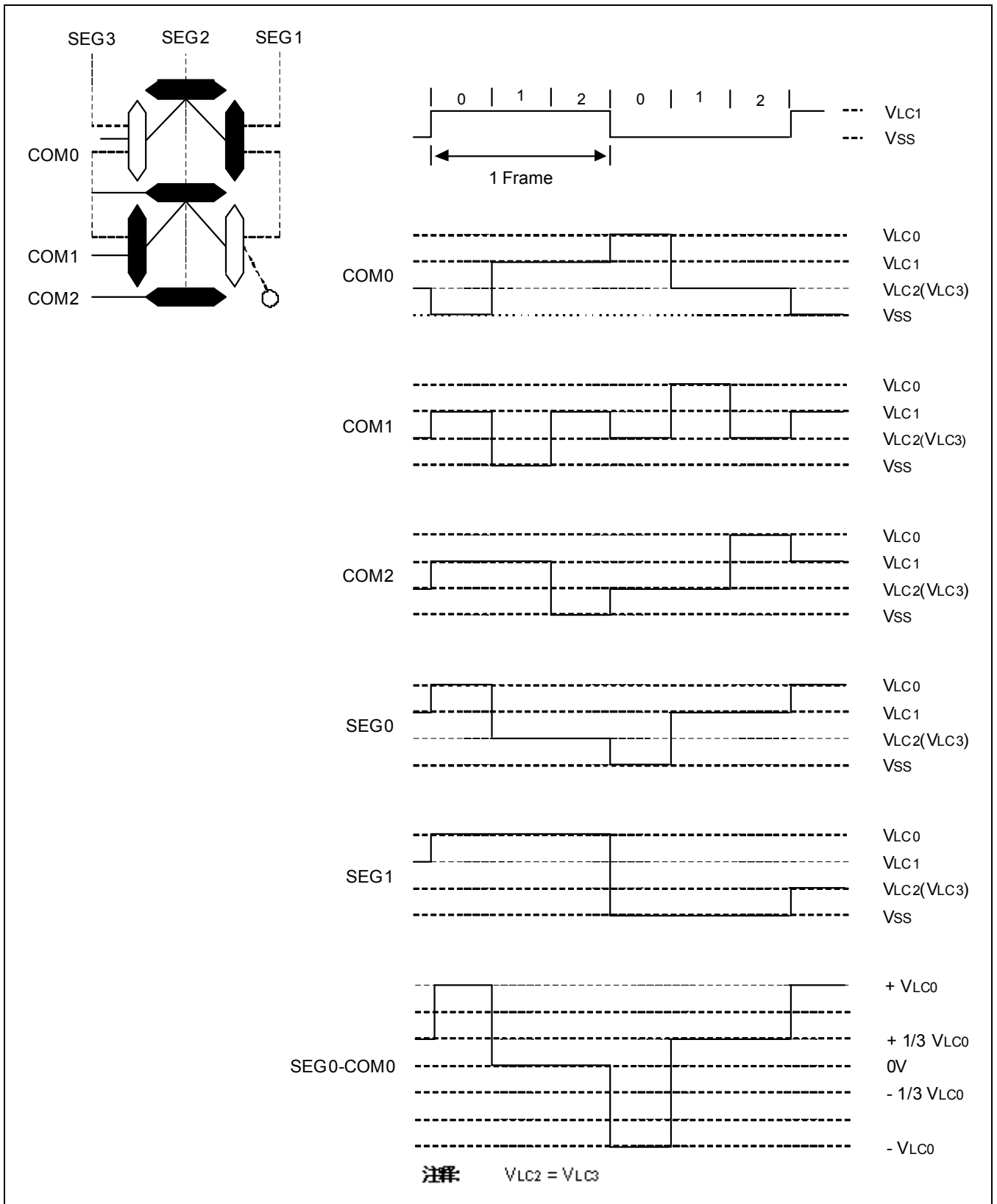


图 14-8 LCD 信号波形 (1/3 占空比, 1/3 偏压比)

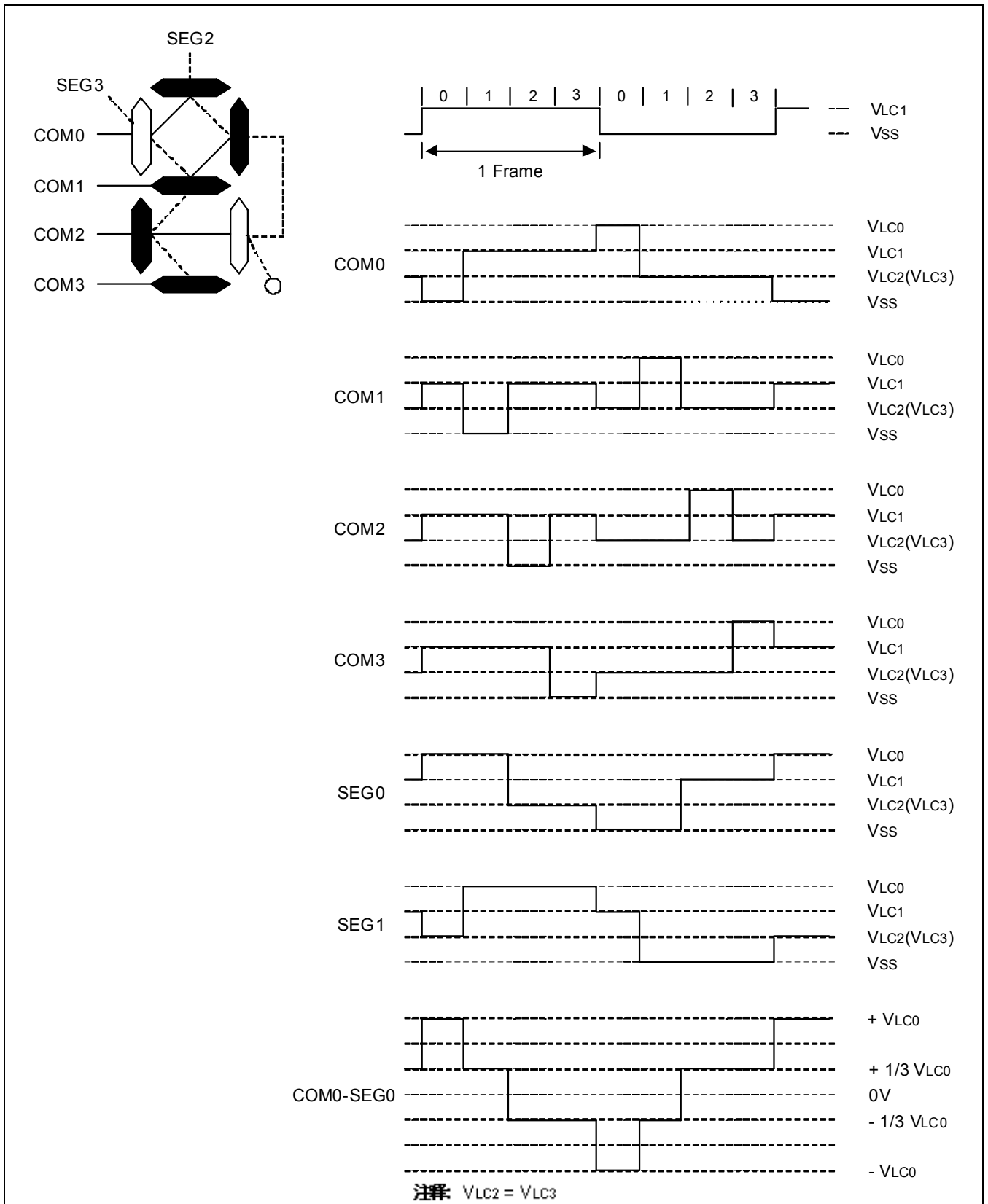


图 14-9 LCD 信号波形 (1/4 占空比, 1/3 偏压比)

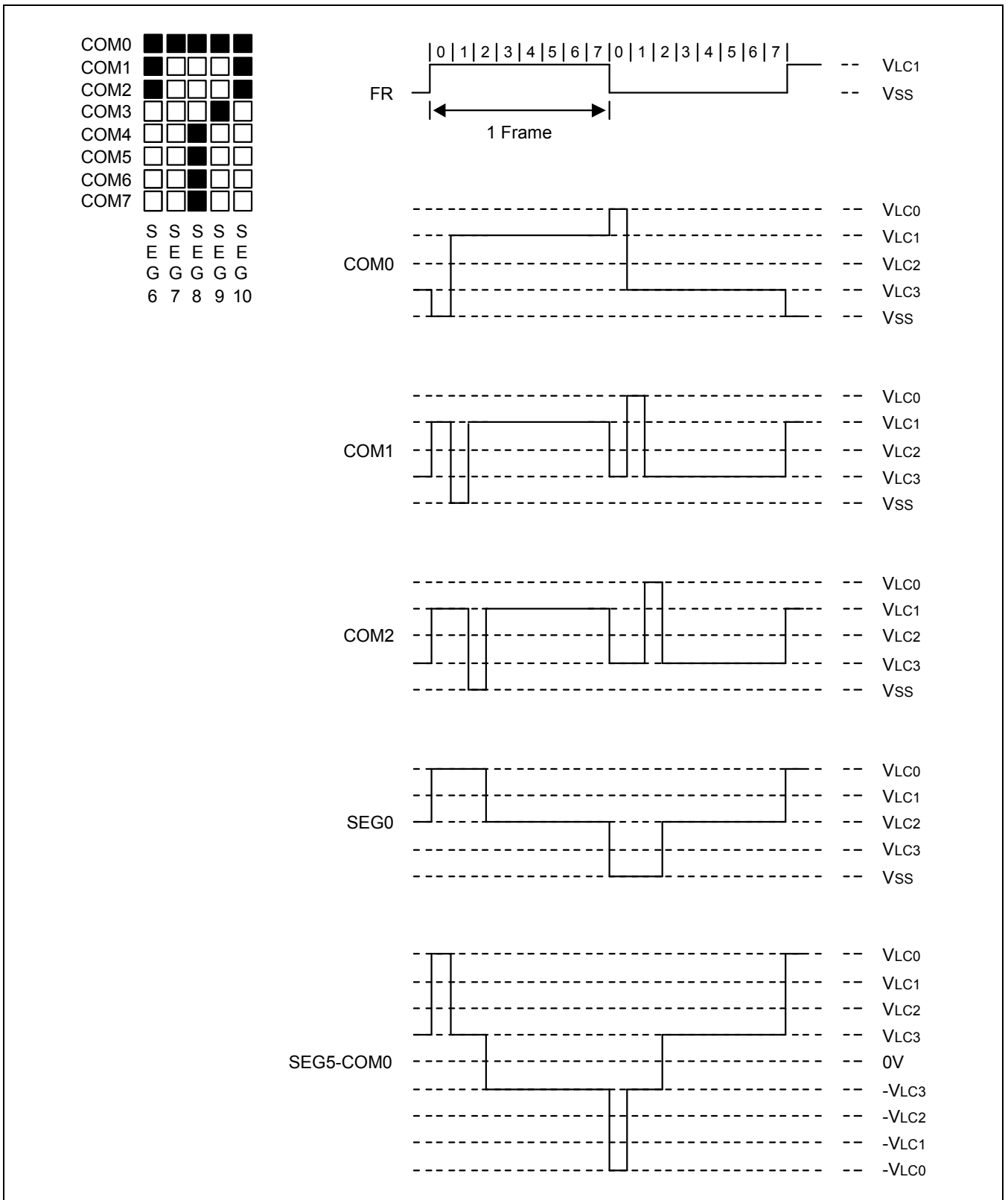


图 14-10 LCD 信号波形 (1/8 占空比, 1/4 偏压比)

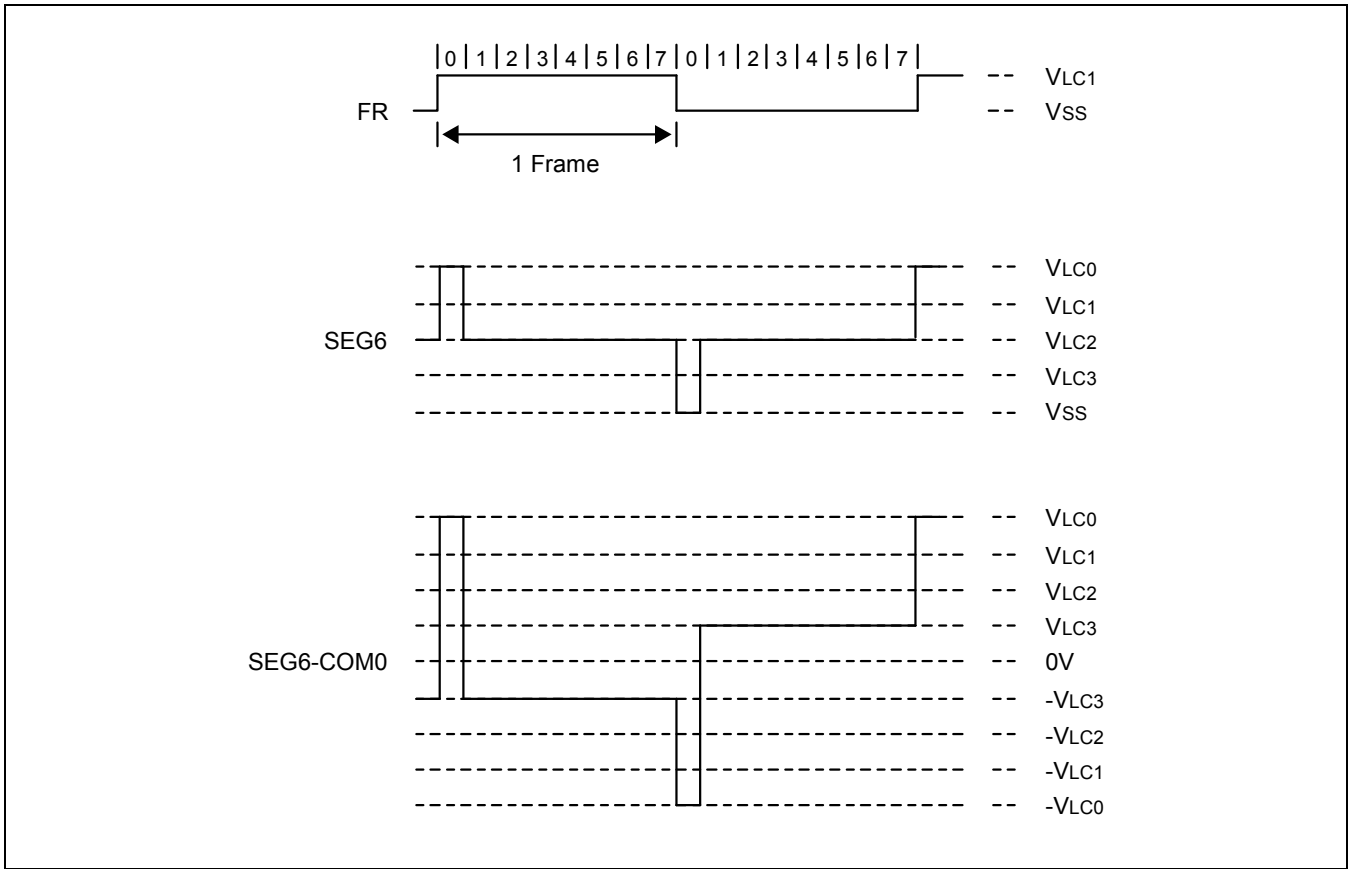


图 14-10. LCD 信号波形 (1/8 占空比, 1/4 偏压比) (续)

# 15

## 10 位 AD 转换器

### 15.1 概述

内部集成了一个8通道10位 A/D 转换器（ADC），通过逐次逼近逻辑把模拟信号转换为相应的10位数字量。模拟输入量的值必须在  $AV_{REF}$  和  $V_{SS}$  之间。A/D 转换器由以下几个部分组成：

- 逐次逼近型模拟比较器
- D/A 转换逻辑(电阻串型)
- AD 转换控制寄存器 (ADCON)
- 8 路模拟信号输入端 (AD0–AD7)
- 10 位 A/D 转换结果寄存器 (ADDATAH/L)
- 8 位数字输入端口 (与 I/O 口复用)
- $AV_{REF}$  和  $AV_{SS}$  管脚， $AV_{SS}$  在芯片内部连接搭到  $V_{SS}$

#### 15.1.1 功能描述

为了开始一次模数转换，首先需要使能 A/D 输入端口，P2 口可以被选择用作 A/D 模拟输入端口。然后写相应值到 A/D 转换控制寄存器 ADCON.4–.6 便可以选择不同的模拟输入通道 (ADC0–7)，最后需要将转换开始位 ADCON.0 置为“1”。A/D 转换控制寄存器的地址为 F3H，set 1, bank0。未使用为 AD 输入的管脚可用作普通 I/O 口。

在一个正常转换过程中，AD 转换逻辑电路先将逐次逼近寄存器的值设为 200H（10 位满量程转换结果的一半）。随后这个寄存器在每次转换时会自动更新。逐次逼近电路每次只能实现一个通道的10位AD转换，但是可以通过操作 ADCON 寄存器的ADCON.6-.4动态分时选择多路 A/D 输入。设置使能位 ADCON.0 位为“1”，则启动 A/D 转换。当 A/D 转换结束时，控制寄存器 ADCON.3 自动置“1”，即 EOC 被置“1”。A/D 转换结果保存于 ADDATAH/L 寄存器，A/D 转换电路随即进入空闲状态。在开始下一次转换前，上一次 A/D 转换的数据必须被读出。否则，前面的结果将被下一次的转换结果覆盖。

**注释：** ADC 电路内部没有采样保存电路，所以在每次 A/D 转换时，此路模拟信号输入端的波动要非常小，否则，转换结果可能会不正确。如果在 A/D 转换期间芯片进入 STOP 或者 IDLE 模式，A/D 模块将会产生漏电流。必须在 A/D 转换结束后再使用 STOP 或者 IDLE 模式。



### 15.1.2 转换时间

A/D 转换 1 位需要 4 个 ADC 时钟周期，建立 A/D 转换需要 10 个时钟周期，因此 A/D 转换完 10 位一共需要 50 个时钟周期。在 8MHz 频率下，一个时钟周期为 1 us (8/f<sub>xx</sub>)。则 A/D 转换所需要的时间为：

**4 时钟周期/位 × 10 位 + 建立时间 (10 个时钟周期) = 50 个时钟周期, 50 个时钟周期 × 1us = 50μs at 1 MHz**

### 15.1.3 A/D 转换控制寄存器 (ADCON)

A/D 转换控制寄存器 ADCON 的地址为 F3H, set 1, bank 0, 该寄存器可以实现以下3个功能:

- 选择模拟输入通道 (ADCON.6-4)
- 转换结束状态检测 (ADCON.3)
- 选择转换速度 (ADCON.2-1)
- 启动 A/D 转换 (ADCON.0)

复位后，转换起始位被禁止。一次只能转换一路模拟量。可以通过操作 ADCON 寄存器的 ADCON.6-4 动态分时选择多路 A/D 输入。未使用为 AD 输入的管脚可用作普通 I/O 口。

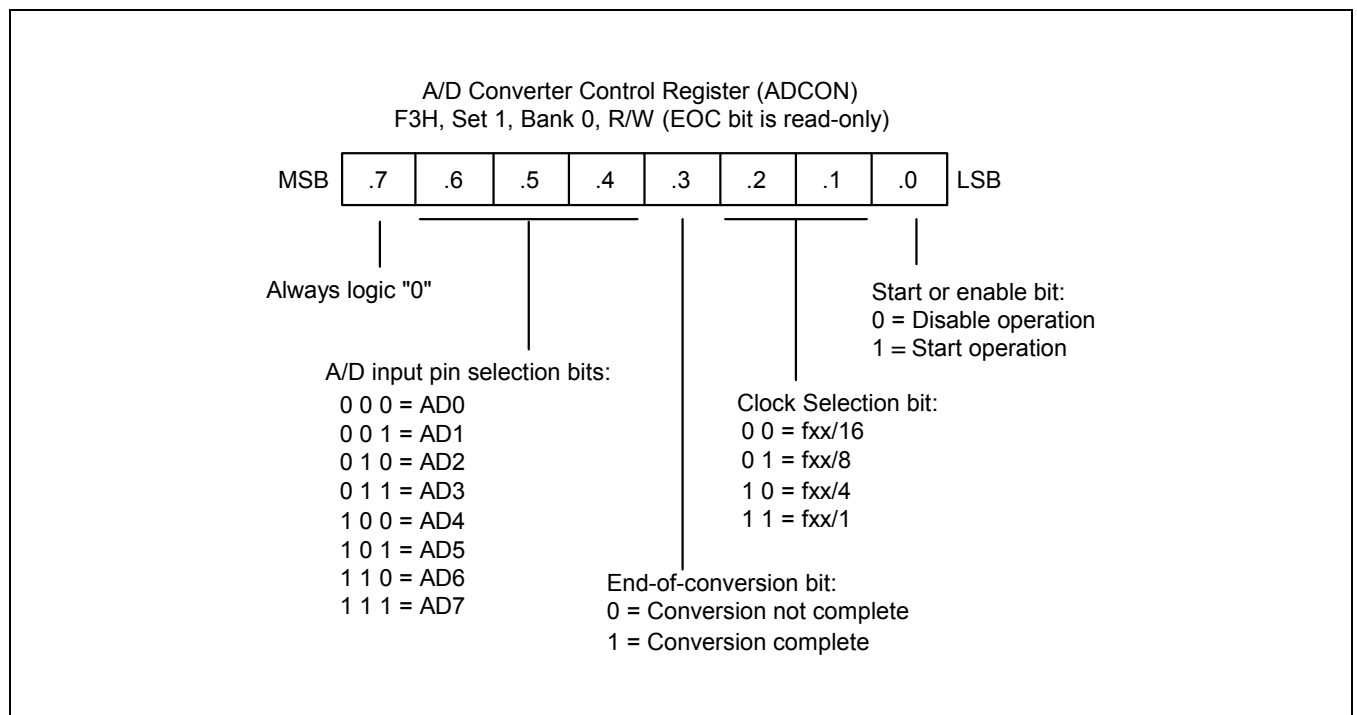


图 15-1 A/D 转换控制寄存器 (ADCON)

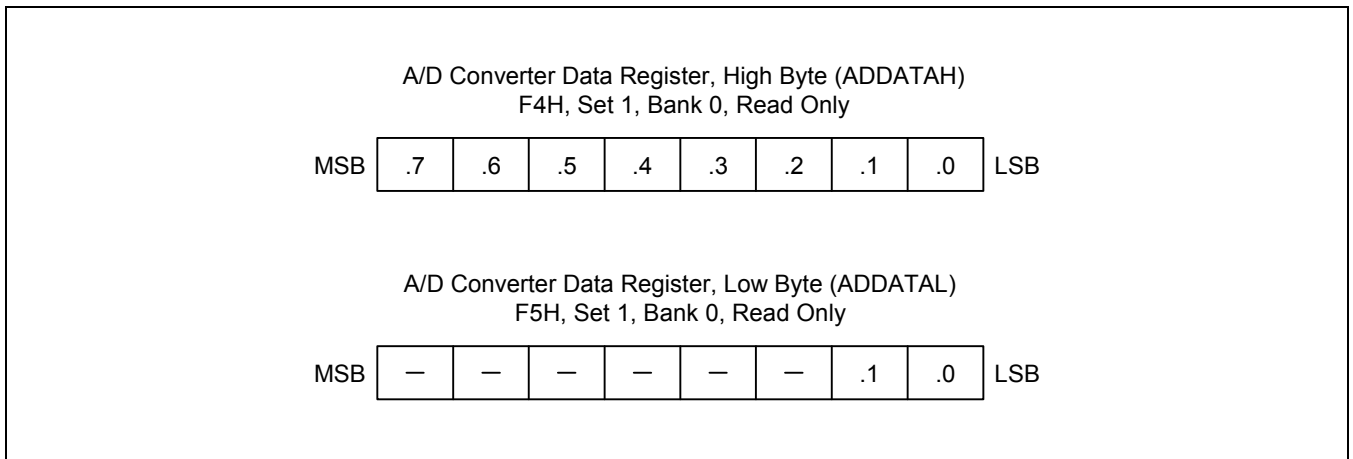


图 15-2 A/D 转换数据寄存器 (ADDATAH/L)

#### 15.1.4 内部参考电压

在 A/D 转换中，输入模拟信号要与内部参考电压进行比较。输入模拟信号的电压范围应在  $AV_{SS}$  至  $AV_{REF}$  之间。(通常,  $AV_{REF} \leq V_{DD}$ )。

不同的比较电压是由内部电阻网络分压产生的，开始的比较电压一般为  $1/2 AV_{REF}$ 。

15.1.5 模块图

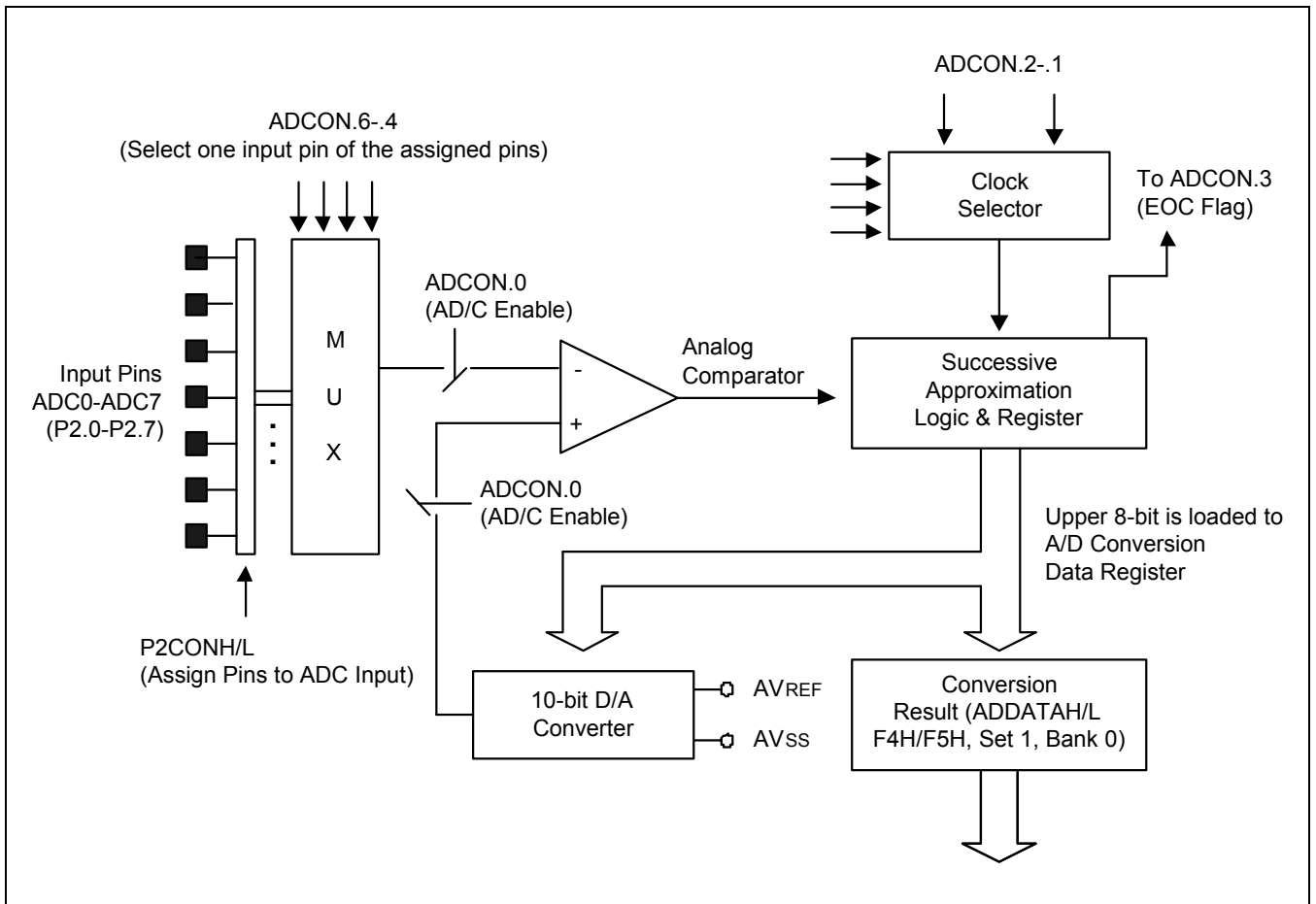


图 15-3 A/D 转换功能模块图

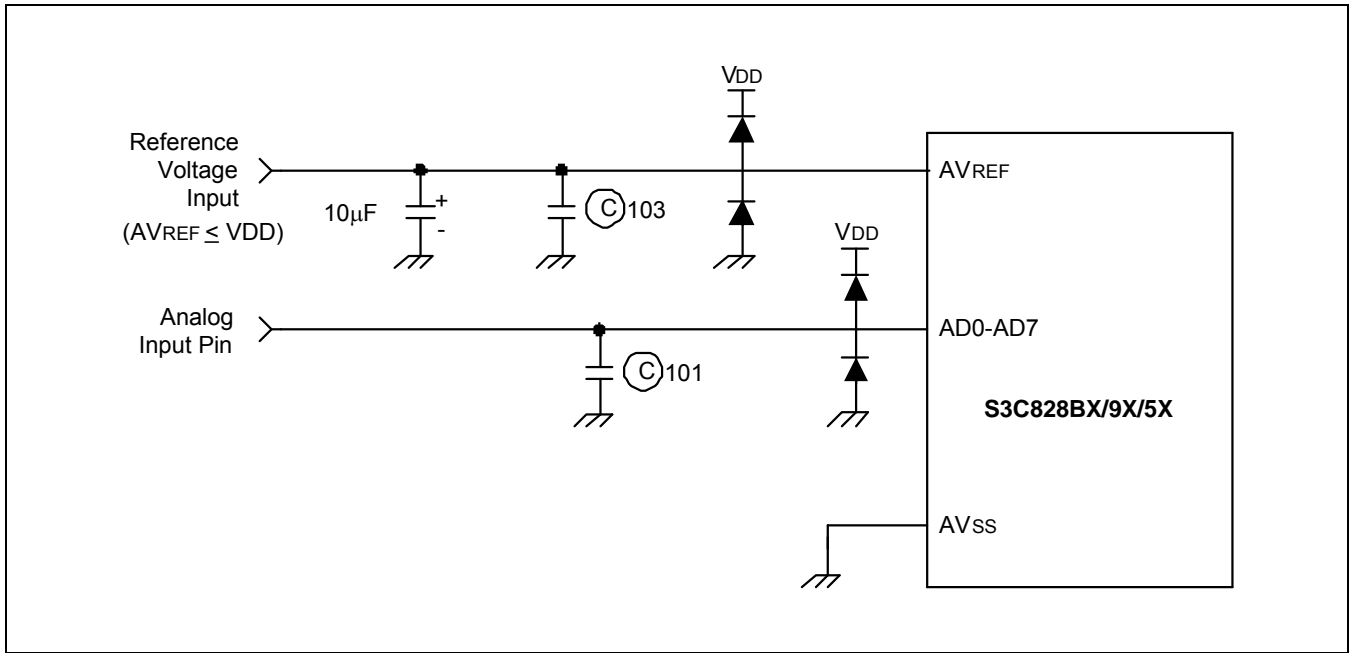


图 15-4 推荐高精度 A/D 转换电路

# 16

## 串行输入输出接口

### 16.1 概述

串行输入输出接口(SIO) 可以与各种外设进行串行数据通讯。SIO 模块主要包括:

- 8 位控制寄存器(SIOCON)
- 时钟选择逻辑
- 8 位数据缓冲器 (SIODATA)
- 8 位预分频器
- 3 位串行时钟计数器
- 串行数据输入/输出引脚 (SI, SO)
- 外部时钟输入引脚 (SCK)

SIO 模块可以接收和发送8位数据，其接收、发送速率由寄存器设定。为了实现灵活的数据传输速率，可以选择内部或者外部时钟源。

#### 16.1.1 编程流程

SIO 模块的编程遵循以下基本的步骤:

1. 通过设置寄存器P1CONH寄存器，设置port 1中SIO对应的I/O口为SO, SCK, SI
2. 设置寄存器SIOCON，正确的配置SIO模块。SIOCON.2 必须设置为'1'以使能数据的移位。
3. 需要中断时，将中断使能位 (SIOCON.1) 置为 '1'。
4. 发送数据时，将要发送的数据写到数据缓冲器 SIODATA，并将 SIOCON.3 设为'1'，则开始发送数据。
5. 当数据接收/发送完成后，SIO中断标志位(SIOCON.0)置为'1'，同时产生SIO中断请求。

### 16.1.2 SIO 控制寄存器 (SIOCON)

SIO 控制寄存器，SIOCON，地址为E0H，Set 1，Bank 0。设置SIO模块的以下功能：

- 时钟源选择 (内部或者外部)
- 中断使能控制
- 移位操作的边沿选择
- 清除3位计数器并开始移位操作
- 使能移位操作(发送)
- 模式选择(发送/接收 或 只接收)
- 数据方向选择 (高位在前或低位在前)

复位后，SIOCON的初始值为'00H'，默认SIO选择内部时钟，工作在仅接收模式下，3位时钟计数器清零。禁止数据移位操作(发送)和中断，选择数据方向为高位在前。

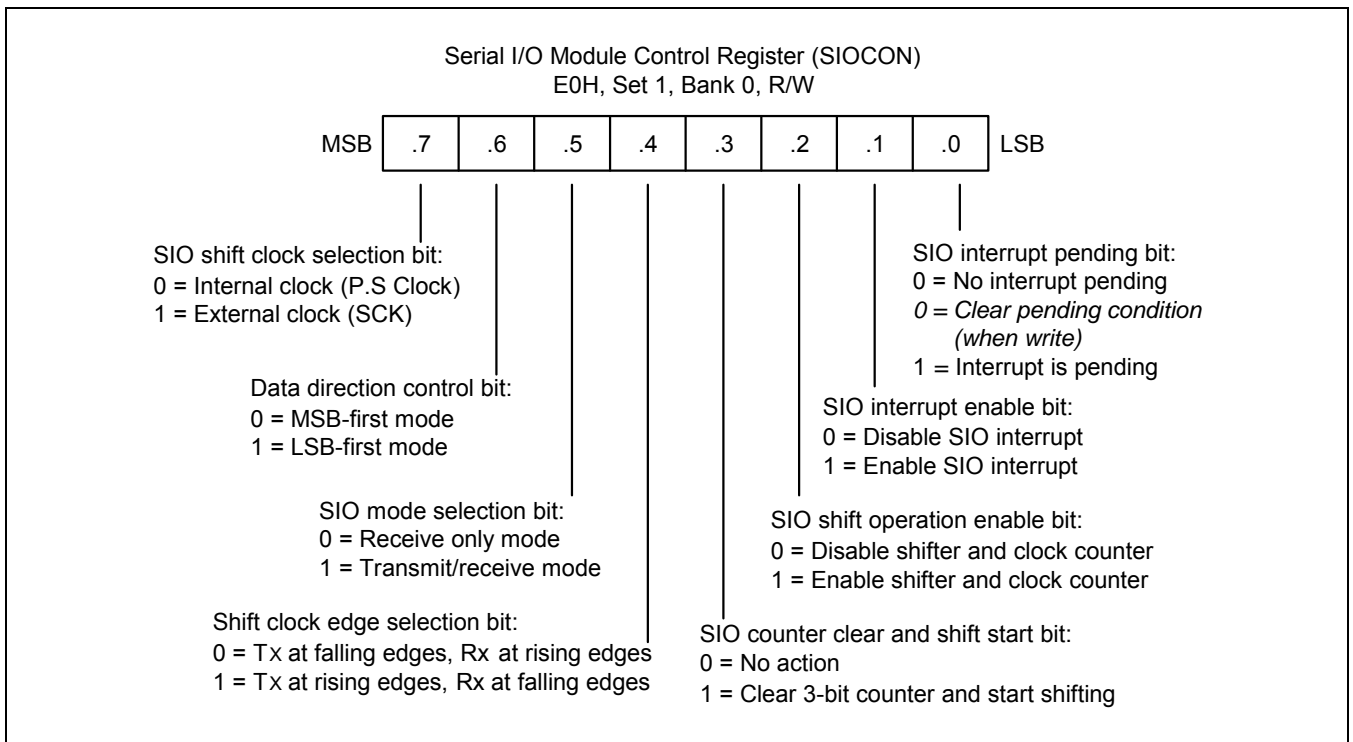


图 16-1 SIO 控制寄存器(SIOCON)

### 16.1.3 SIO 预分频寄存器 (SIOPS)

SIO 预分频寄存器, SIOPS, 地址为 E2H, Set 1, Bank 0。SIOPS寄存器的值决定了SIO数据传输的波特率:

波特率 = 输入时钟频率( $f_{xx}/4$ ) / (预分频值+ 1), 或外部输入时钟频率, 这里输入时钟为 $f_{xx}/4$ 。

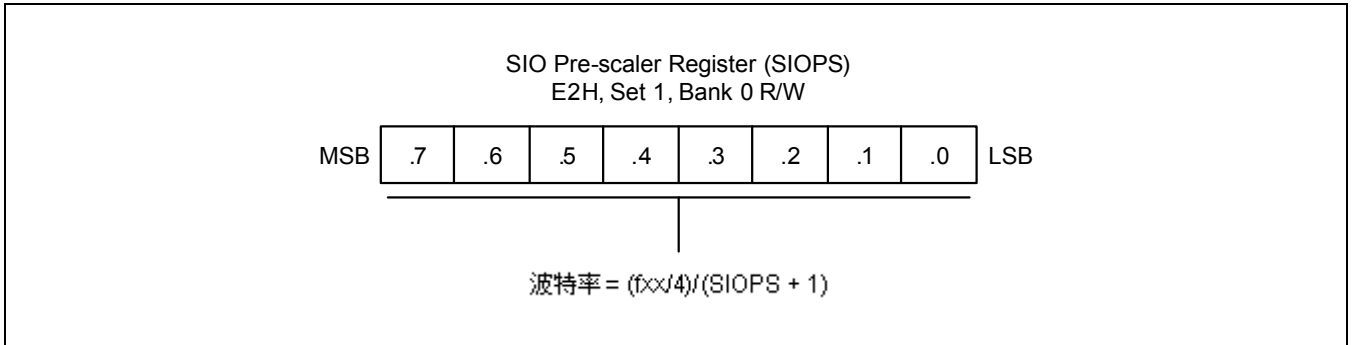


图 16-2 SIO 预分频寄存器 (SIOPS)

### 16.1.4 模块框图

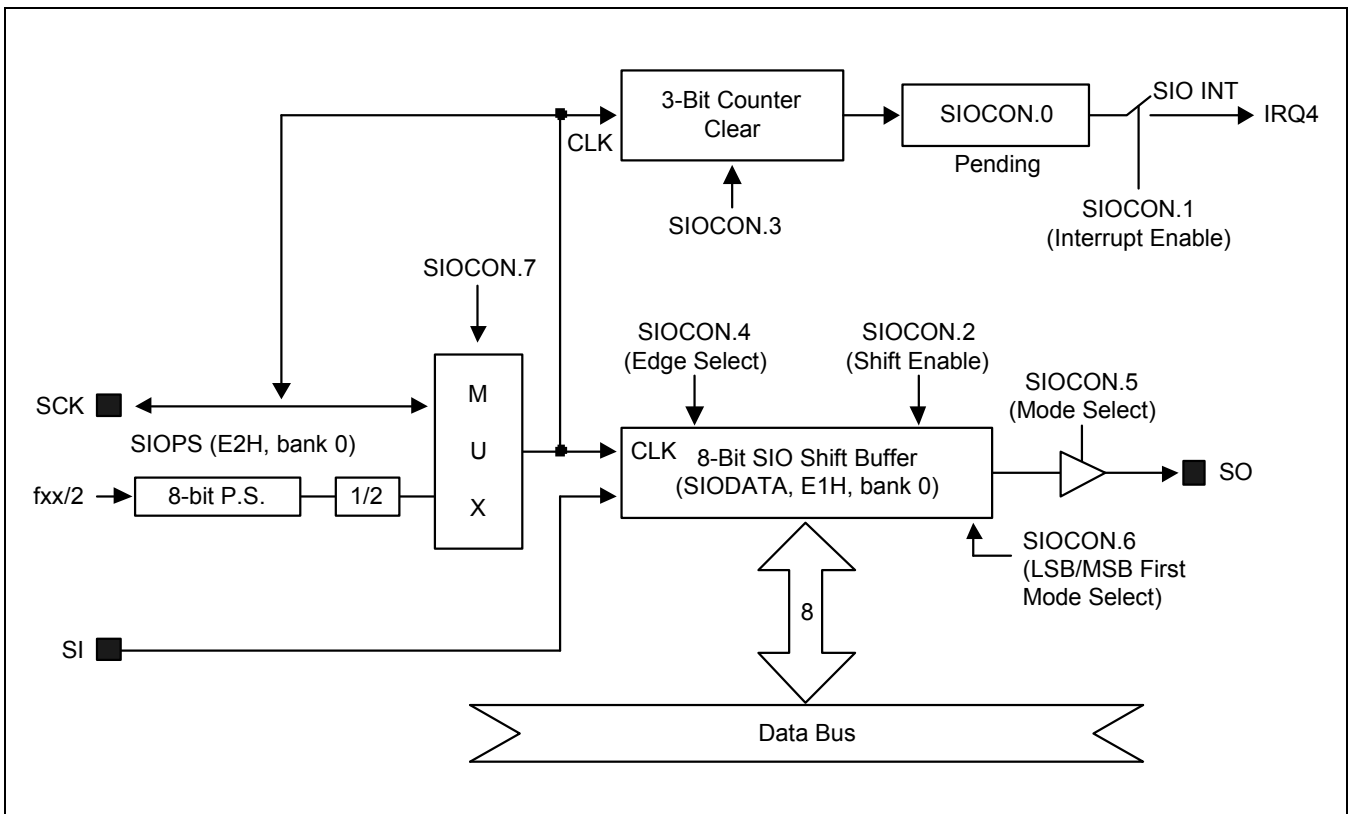


图 16-3 SIO 功能模块框图

16.1.5 SIO 时序图

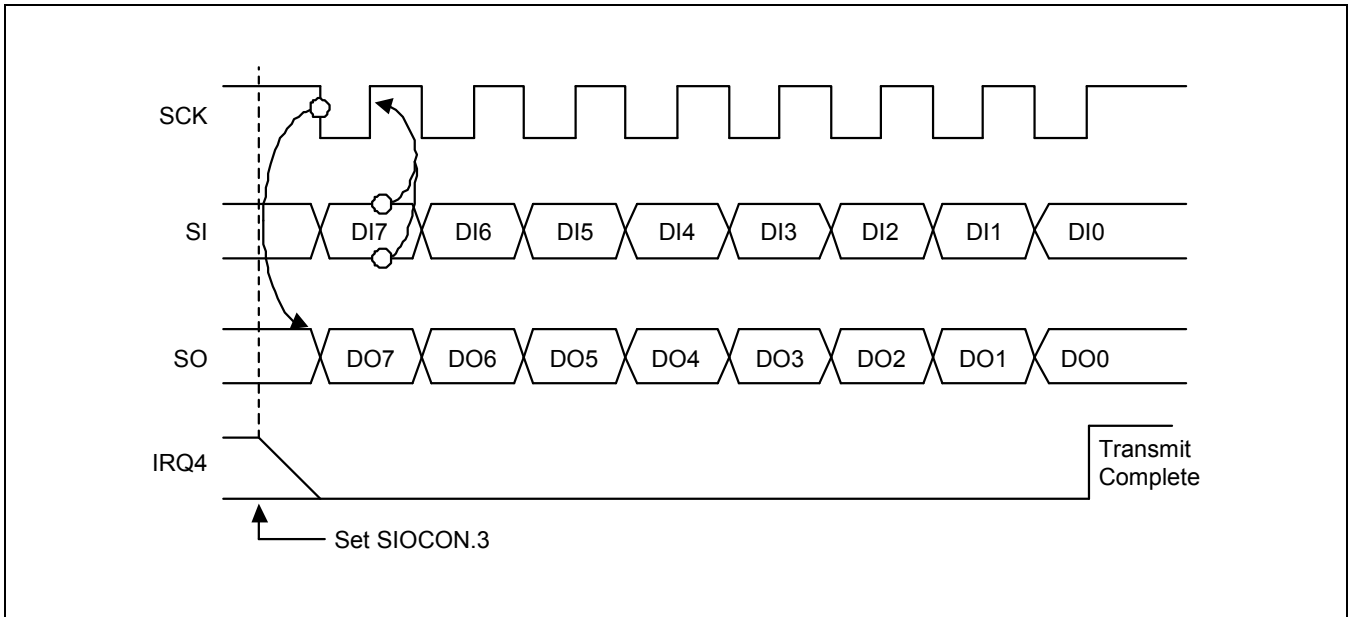


图 16-4 SIO 串行发送-接收模式 (下降沿发送, SIOCON.4 = 0)

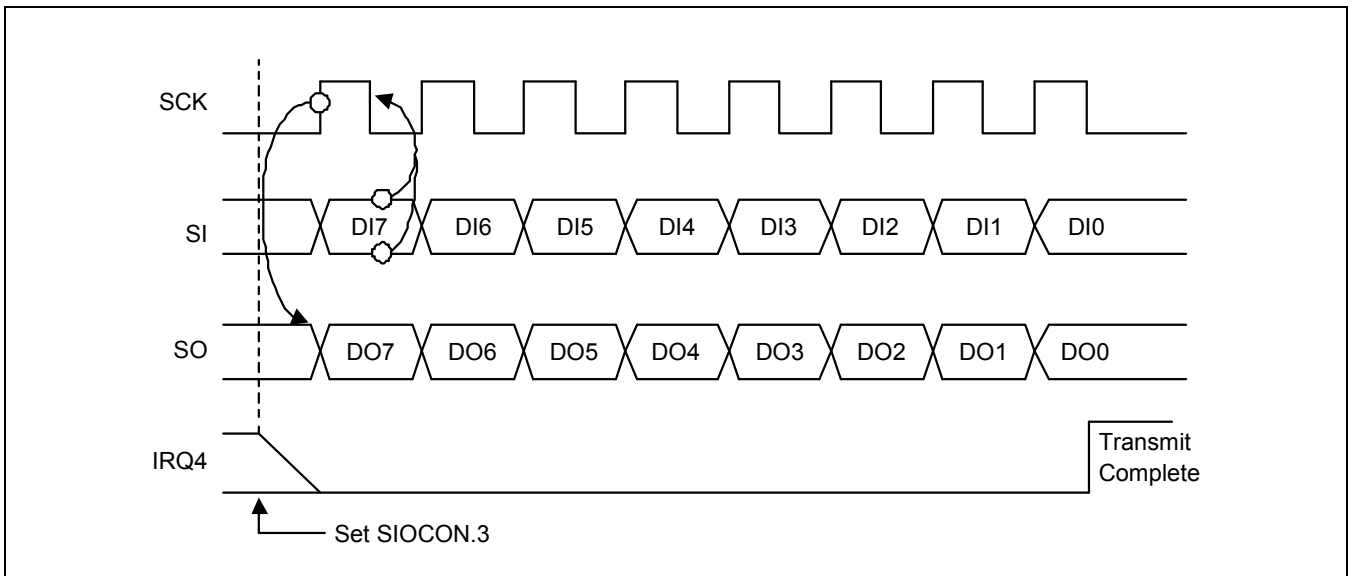


图 16-5 SIO 串行发送-接收模式 (上升沿发送, SIOCON.4 = 1)



# 17

## UART

### 17.1 概述

UART 模块是运行模式可设置的全双工串行通讯口。

其运行模式包括一个并行模式和三个UART(通用异步串行接收/发送)模式:

- 串行I/O口, 波特率为  $fx / (16 \times (BRDATA+1))$
- 8位 UART 模式, 波特率可设置
- 9位 UART 模式, 波特率固定为:  $fx / 16$
- 9位 UART 模式, 波特率可设置

UART的接收和发送都是通过UART数据寄存器 UDATA(F7H, Set 1, Bank 0) 来实现的。

写UART数据寄存器时, 数据装载到UART的发送缓冲器中; 读UART数据寄存器时, 数据从UART的接收缓冲器中读出。发送缓冲器和接收缓冲器在物理上是分开的。

接收缓冲器为移位寄存器, 在上一个接收的数据还未读出前, 下一个数据已经开始接收。因此, 如果在下一个数据完成接收之前, 上一个接收的数据还没有读出的话, 则上一个数据会丢失(过载错误)。

在所有运行模式下, 当任何指令将UDATA作为目标地址写入数据时, 则立即开始发送。在mode 0 模式下, 只有当接收中断标志位(INTPND.5)为‘0’和接收使能位(UARTCON.4)为‘1’时, 才开始接收串行数据。在mode 1 和 mode 2模式下, 当接收使能位(UARTCON.4)为‘1’时, 任何时刻只要收到数据起始位(“0”)则立即开始接收数据。

#### 17.1.1 编程流程

UART模块编程的基本步骤为:

1. 通过设置P3CONH为相应的值, 将P3.5和P3.4设为UART功能引脚(RXD (P3.5), TXD (P3.4))。
2. 设置UARTCON控制寄存器来选择UART的运行模式。
3. 需要中断时, 将UART中断使能位(UARTCON.1 或 UARTCON.0)设为‘1’。
4. 需要传送数据时, 将需要传送的数据写入到UDATA, 则开始传送。
5. 当发送/接收完一个数据后, UART 标志位 (INTPND.4 或 INTPND.5)被置为‘1’, 同时产生一个相应的发送/接收中断。

### 17.1.2 UART 控制寄存器(UARTCON)

UART的控制寄存器是UARTCON，地址为F6H, Set 1, Bank0. 主要实现以下功能：

- 运行模式和波特率选择
- 多节点通讯和中断控制
- 串行接收使能/禁止控制
- Mode 2 和mode 3 模式下发送和接收时第9位数据位的位置
- UART 发送和接收中断控制

复位后，UARTCON的值被清为“00H”，因此，如果需要使用UART模块，应设定UARTCON为适当的值。

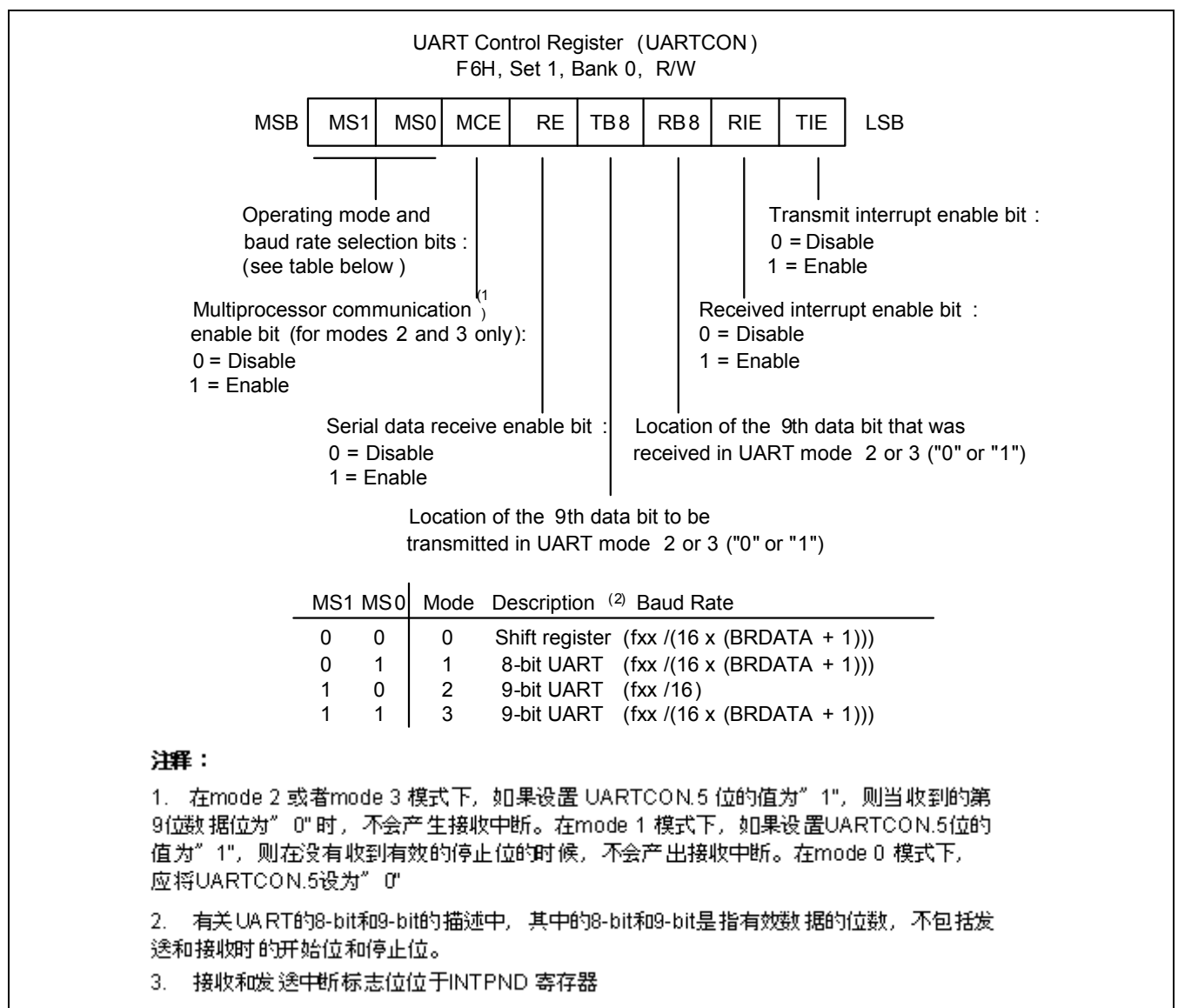


图 17-1 UART 控制寄存器 (UARTCON)

### 17.1.3 UART 中断标志位

UART中断标志位，INTPND.5–.4，位于F9H, set 1, bank 0。

包括UART数据发送中断标志位(INTPND.4)和接收中断标志位(INTPND.5)。

在mode 0 模式下，当收到第8位数据位后，接收中断标志位INTPND.5被置为"1"。在mode 1模式下，INTPND.5在停止位的中间点置"1"。在mode 2 或 mode 3模式下，在接收到 RB8 位的中间点时，INTPND.5置"1"。当CPU响应了接收中断后，INTPND.5必须在中断服务程序中软件清零。

在UART 的mode 0 模式下，当发送完第8位数据后，发送中断标志位INTPND.4被置为"1"。在mode 1 ， mode 2 或mode 3 模式下，INTPND.4在停止位开始发送时置"1"。当CPU响应了发送中断后，INTPND.4必须在中断服务程序中软件清零。

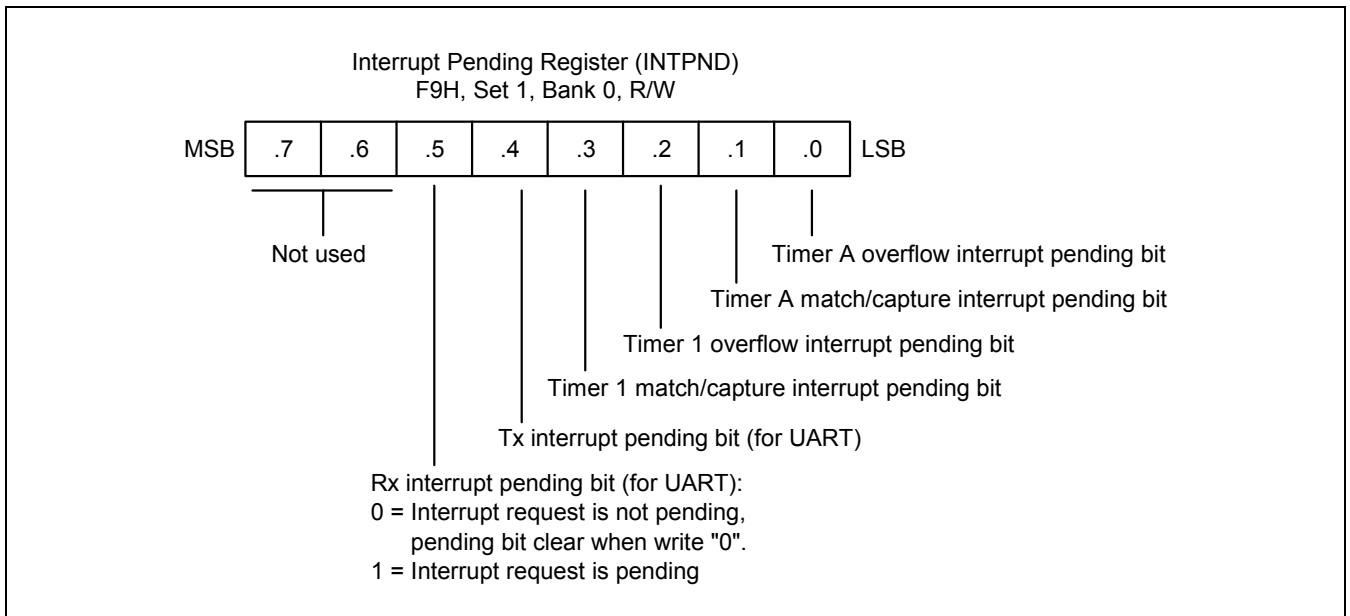


图 17-2 UART 中断标志位 (INTPND.5–.4)

### 17.1.4 UART 数据寄存器 (UDATA)

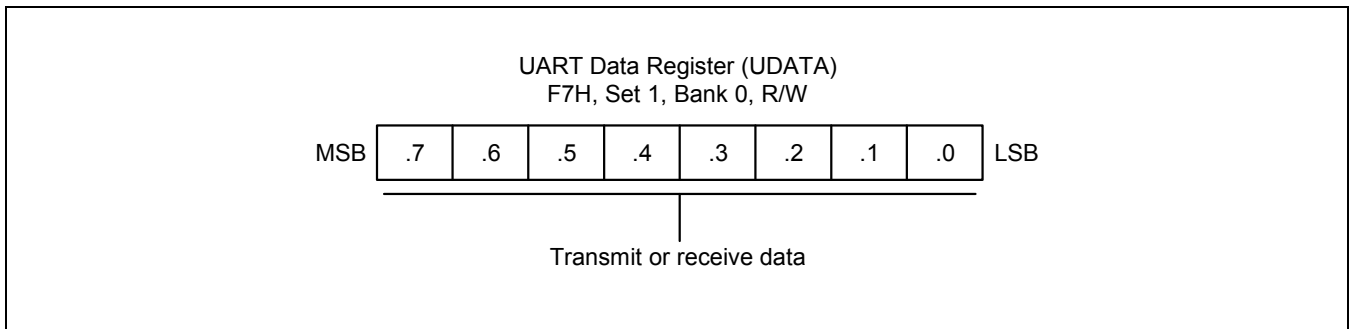


图 17-3 UART 数据寄存器 (UDATA)

### 17.1.5 UART 波特率数据寄存器 (BRDATA)

通过设置UART波特率数据寄存器BRDATA，可以设定UART的时钟速率(即波特率)。

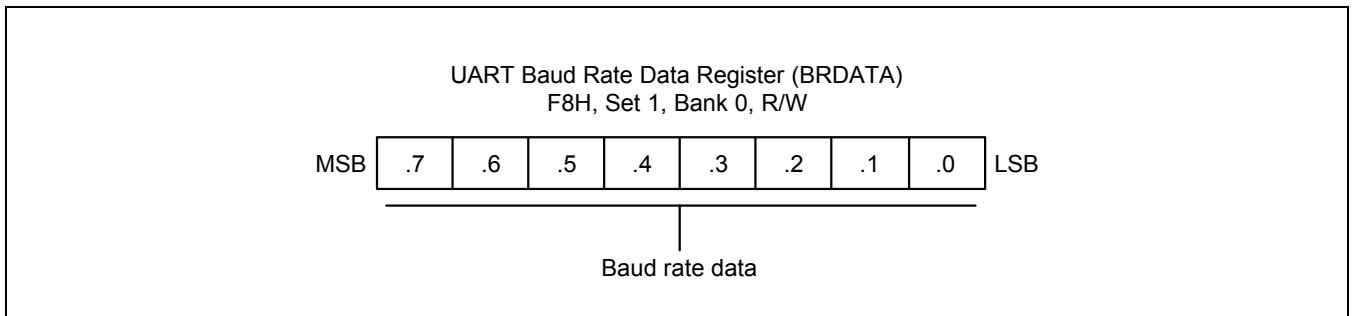


图 17-4 UART 波特率数据寄存器(BRDATA)

### 17.1.6 波特率计算

#### 17.1.6.1 Mode 0 模式下波特率的计算

在mode 0 模式下，波特率由波特率数据寄存器，BRDATA (F8H, Set 1, bank 0)的值决定，mode 0模式下波特率为： $f_U / (16 \times (BRDATA + 1))$

#### 17.1.6.2 Mode 2 模式下波特率的计算

Mode 2 模式下的波特率固定为  $f_{xx}/16$ ，即 $f_{xx}$ 信号16分频。

#### 17.1.6.3 Mode 1 和mode 3 模式下波特率的计算

在mode 1和mode 3模式下，波特率由波特率数据寄存器，BRDATA (F8H, Set 1, bank 0)的值决定，mode 1和mode 3模式下波特率为： $f_U / (16 \times (BRDATA + 1))$

表 17-1 8-位 BR0DATA可以产生的常用波特率表

模式	波特率	时钟频率	BRDATA	
			十进制	十六进制
Mode 2	0.5MHz	8MHz	x	x
Mode 0	230,400Hz	11.0592MHz	02	02H
Mode 1	115,200Hz	11.0592MHz	05	05H
Mode 3	57,600Hz	11.0592MHz	11	0BH
	38,400Hz	11.0592MHz	17	11H
	19,200Hz	11.0592MHz	35	23H
	9,600Hz	11.0592MHz	71	47H
	4,800Hz	11.0592MHz	143	8FH
	62,500Hz	10MHz	09	09H
	9,615Hz	10MHz	64	40H
	38,461Hz	8MHz	12	0CH
	12,500Hz	8MHz	39	27H
	19,230Hz	4MHz	12	0CH
	9,615Hz	4MHz	25	19H

17.1.7 模块框图

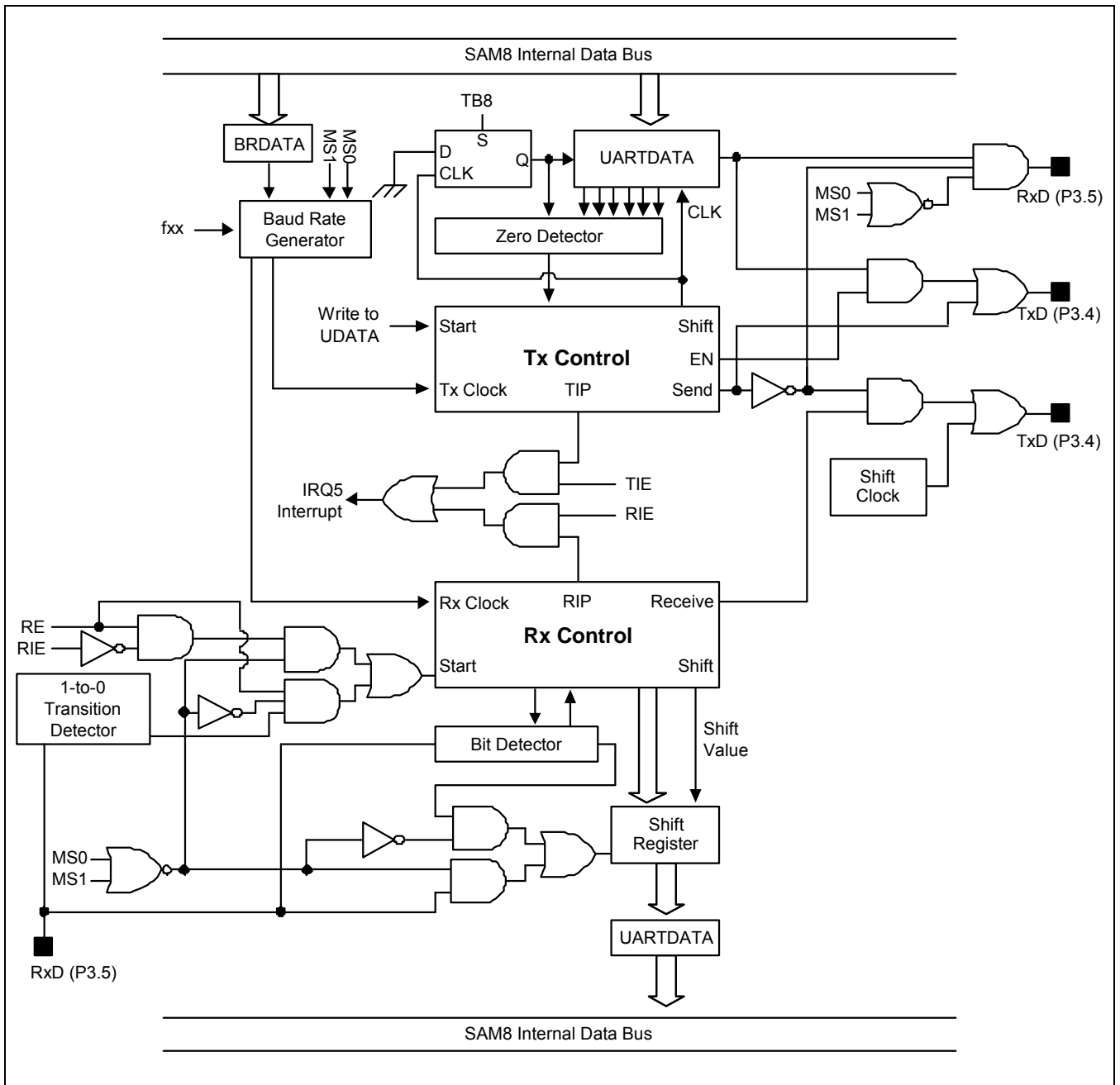


图 17-5 UART 功能模块框图

### 17.1.8 UART MODE 0 模式功能描述

在mode 0 模式下，UART通过RxD(P3.5)接收和发送数据，TxD(P3.4) 输出数据移位时钟信号。数据接收和发送的长度都是8位，首先发送或接收的是数据的低位(LSB)。

#### 17.1.8.1 Mode 0 模式数据发送流程：

1. 设置UARTCON.7-.6为"00B"，选择mode 0。
2. 将需要发送的数据写入UDATA (F7H, set 1, bank 0)，数据开始发送。

#### 17.1.8.2 Mode 0 模式数据接收流程：

1. 设置UARTCON.6-.7为"00B"，选择mode 0。
2. 写"0"到INTPND.5，清除接收中断标志位。
3. 设置UART接收使能位(UARTCON.4)为"1"，使能UART接收。
4. 时钟移位信号开始输出到TxD (P3.4)，同时开始从RxD (P3.5) 读取数据。当UARTCON.1置为"1"时，产生UART接收中断(向量地址:IRQ5， ECH)。

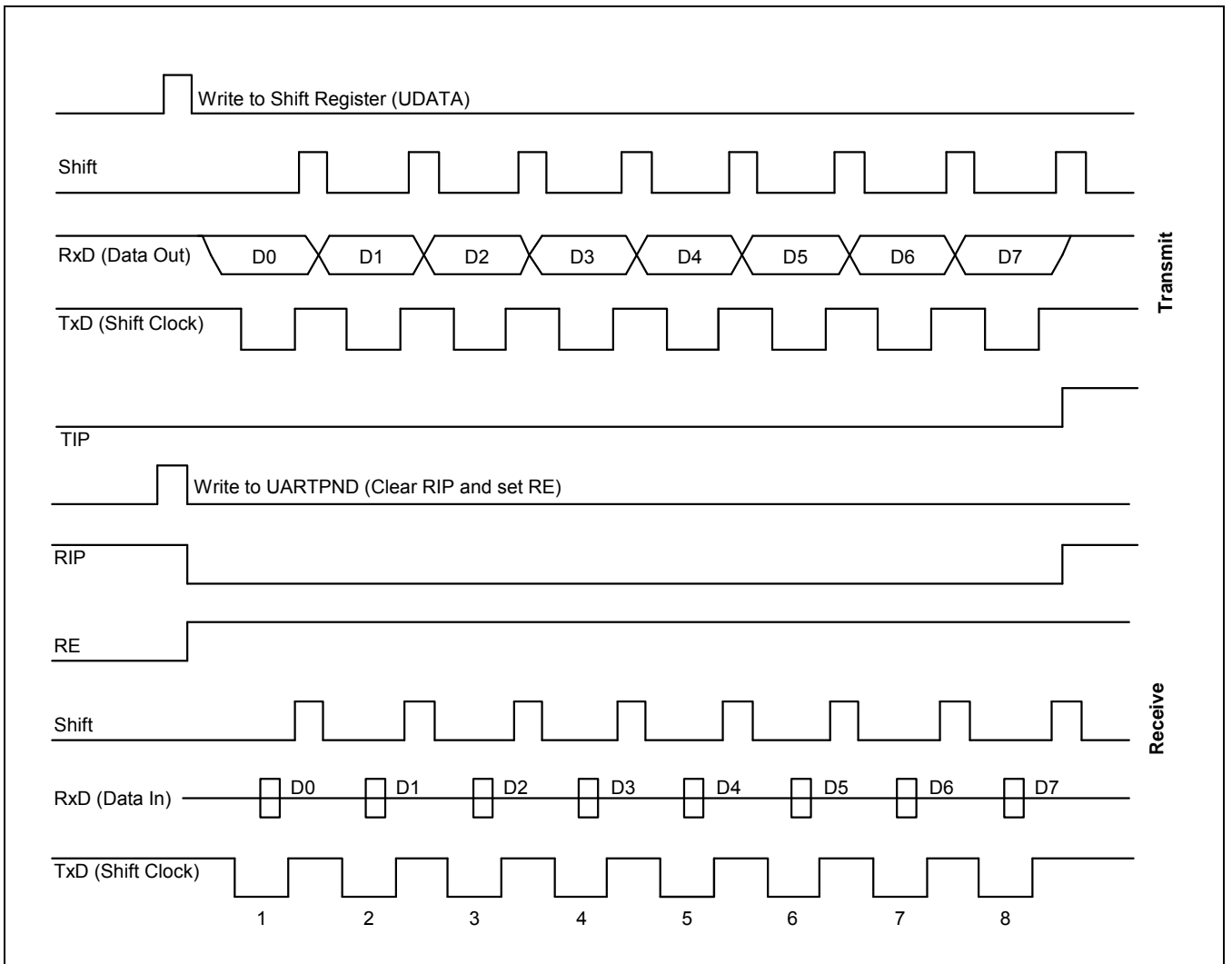


图 17-6 UART Mode 0 模式运行时序图



### 17.1.9 UART MODE 1 模式功能描述

在mode 1 模式下，发送(通过TxD (P3.4)) 或者接收 (通过RxD (P3.5))的数据帧总长度是10位。  
每一个数据帧由三部分组成：

- 起始位 ("0")
- 8位有效数据 (低位在前)
- 停止位 ("1")

Mode 1模式的波特率是可变的。

#### 17.1.9.1 Mode 1 模式数据发送流程

1. 通过设置8 位 BRDATA寄存器设定波特率。
2. 设置UARTCON.7-.6为"01B"，选择mode 1(8位 UART)。
3. 将需要发送的数据写入UDATA (F7H, set 1, bank 0)，数据开始发送。起始位和停止位由硬件自动产生。

#### 17.1.9.2 Mode 1 模式数据接收流程

1. 通过设置8位 BRDATA 寄存器设定波特率。
2. 选择mode 1 模式并设置 UARTCON 的 RE (接收使能) 位为 "1"。
3. 当RxD (P3.5) 管脚上检测到起始条件(低电平"0")时，UART模块开始进行串行数据接收操作。

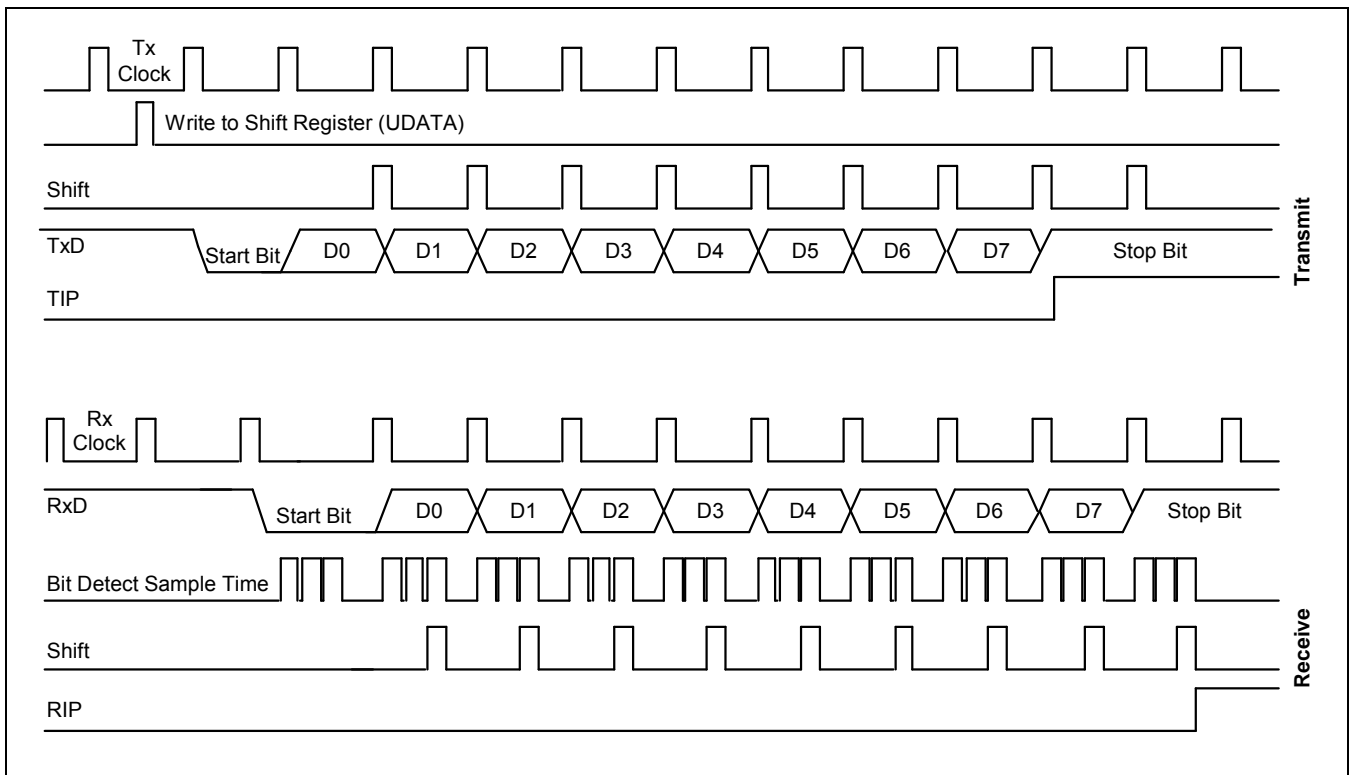


图 17-7 UART Mode 1 模式运行时序图

### 17.1.10 UART MODE 2 模式功能描述

在mode 2 模式下，发送(通过TxD (P3.4)) 或者接收 (通过RxD (P3.5))的数据帧总长度是11位。  
每一个数据帧由三部分组成：

- 起始位("0")
- 8位有效数据 (低位在前)
- 可编程设置的第9位数据位
- 停止位 ("1")

发送时，将需要发送的第9位数据的值"0" 或 "1" 的写入到TB8 位 (UARTCON.3)。

接收时，接收到的第9位数据存放在RB8 位 (UARTCON.2)，同时停止位被忽略。 Mode 2 模式的波特率为  $f_{osc}/16$ 。

#### 17.1.10.1 Mode 2 模式发送数据流程

1. 设置UARTCON.7-6为"10B"，选择 mode 2 (9位 UART)，同时将需要发送的第9位数据写入TB8。
2. 将需要发送的数据写入UDATA (F7H, set 1, bank 0)，数据开始发送。

#### 17.1.10.2 Mode 2 模式数据接收流程

1. 选择mode 2 模式并设置 UARTCON 的 RE (接收使能) 位为 "1"。
2. 当RxD (P3.5) 管脚上的电平变为低时，开始接收数据。

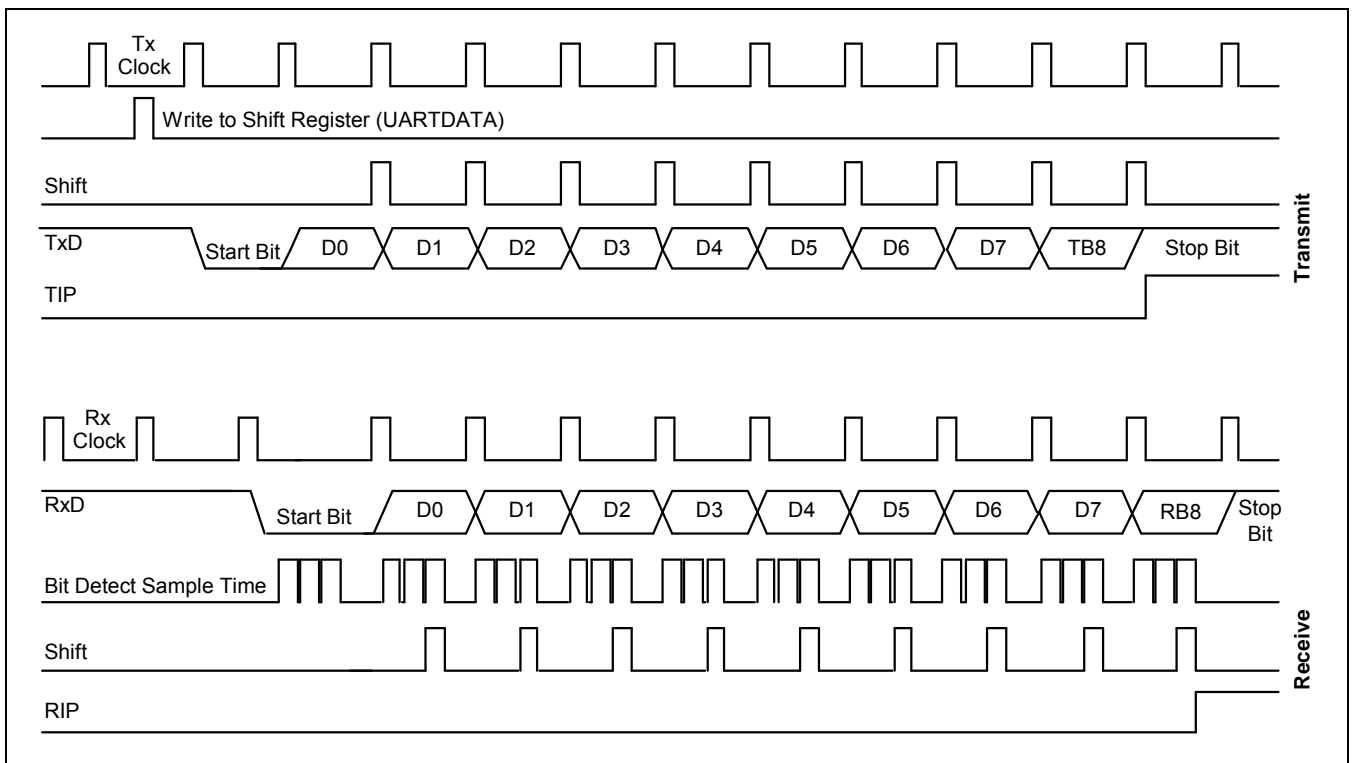


图 17-8 UART Mode 2 模式运行时序图

### 17.1.11 UART MODE 3 模式功能描述

在 mode 3 模式下，发送(通过TxD (P0.1)) 或者接收 (通过RxD (P0.0))的数据帧总长度是11位。除了mode 3 模式的波特率可以调节之外，其余的功能mode 3 和 mode 2 的是完全相同的。每一个数据帧由四部分组成：

- 起始位("0")
- 8位有效数据 (低位在前)
- 可编程设置的第9位数据位
- 停止位 ("1")

#### 17.1.11.1 Mode 3 模式发送数据流程

1. 通过设置8位 BRDATA 寄存器设定波特率。
2. 设置UARTCON.7-.6为"11B", 选择mode 3 (9位 UART)，同时将需要发送的第9位数据写入TB8 (UARTCON.3)。
3. 将需要发送的数据写入UDATA (F7H, set 1, bank 0)，数据开始发送。

#### 17.1.11.2 Mode 3 模式数据接收流程

1. 通过设置8位 BRDATA 寄存器设定波特率。
2. 选择mode 3 模式并设置 UARTCON 的 RE (接收使能) 位为 "1"。
3. 当RxD (P3.5) 管脚上的电平变为低时，开始接收数据。

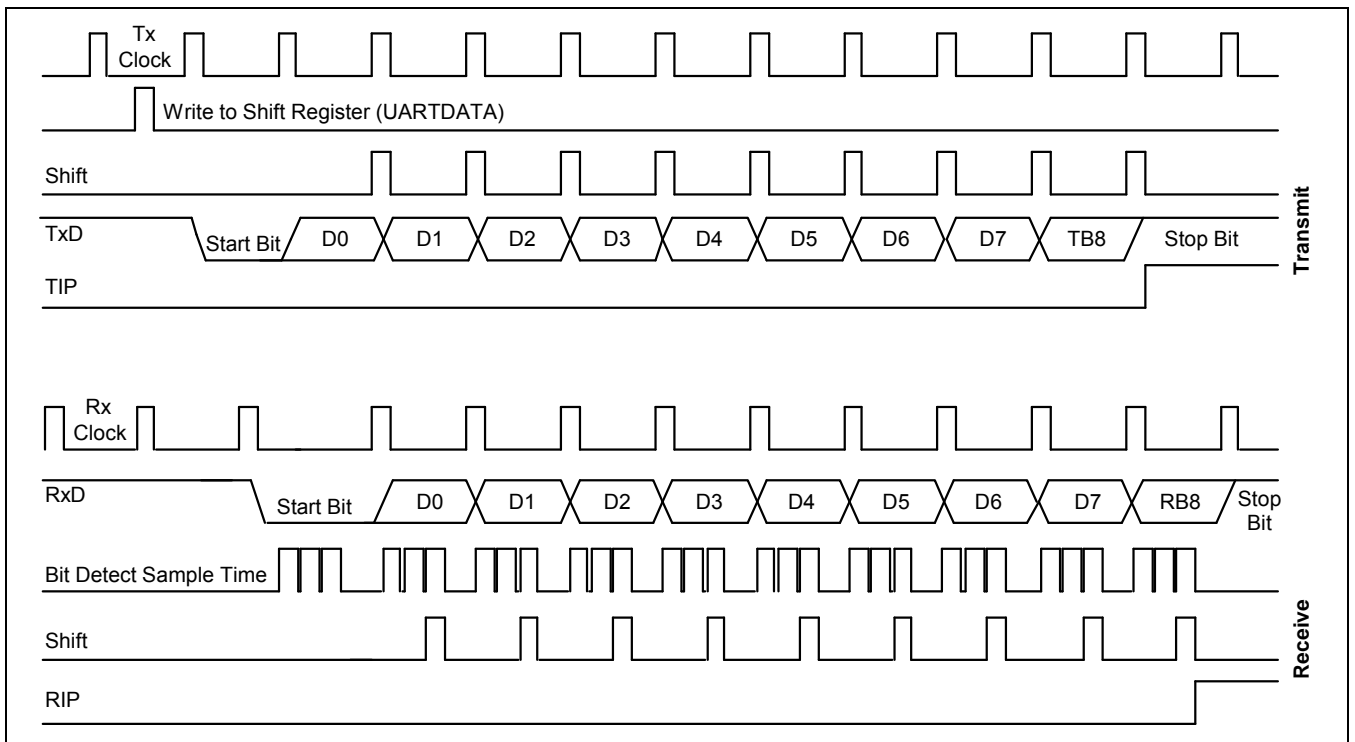


图 17-9 UART Mode 3 模式运行时序图

### 17.1.12 多节点串行通讯

S3F8 系列的多节点通讯功能的特点是：

在一个由多个S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X组成的串行通信系统中，允许一个S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X作为主机向另一个特定的从机设备发送串行数据，而不会影响连接在同一个串行线上的其他从机设备。

这个功能只有在UART 的mode 2 和mode 3 模式下，禁止奇偶校验时才可以实现。在mode 2 和mode 3 模式下，收到的数据供9位，第9位写入了RB8 (UARTCON.2)。

接收操作在收到停止位后完成。因此可以编程实现这样的功能：收到停止位后，只有当 RB8 = "1" 时才产生中断。

为了实现这个功能，需要设置 UARTCON寄存器中的MCE位，当设置MCE位为"1"时，

接收的串行数据帧中的第9位数据 = "0" 时不会产生中断。

在这种情况下，就可以使用第9位来简单的区分地址和数据。

### 17.1.13 主机/从机交互协议的一个例子

当主机希望给串接在同一条串行总线上的很多从机中的一个从机发送数据时，

它首先发送地址信息来选中目标从机。注意在这种情况下，一个地址帧和一个数据帧是不同的，地址帧的第9位数据为"1"，而数据帧的第9位数据为"0"。

地址帧会让所有的从机都产生中断，并被每个从机都接收到，则每个从机就可以通过检查接收到的数据来判断是否被选中。被选中的目标从机会将MCE 位清为"0"并准备开始接收数据。

没有被选中的从机的MCE位继续保持为"1"，保持正常运行，并忽略串行总线上的数据帧。

MCE位在mode 0 模式下没有用处。在mode 1 模式下，它可以用来检测停止位的有效性。在mode 1 模式下接收数据时，如果设置MCE 为"1"，只有接收到的停止位有效(为"1")时，才可以产生接收中断。

### 17.1.14 多节点通讯的建立流程

多节点通讯的建立流程如下：

1. 设置所有的串行总线的S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X设备的 UART 模式为 mode 2 或者 mode 3。
2. 设置所有从机设备的MCE位为"1"。
3. 主机的传输协议为：
  - 第一个字节：地址  
标识目标从机设备地址 (第9位 = "1")
  - 下一个字节：数据  
(第9位 = "0")
4. 由于第9位为"1"，所有的从机都会产生中断来接收第一个字节。当目标从机收到第一个字节后将收到的地址与自己的地址进行比较，确认被选中后，则将MCE位清为"0"来准备接受数据。其他的从机继续正常运行。

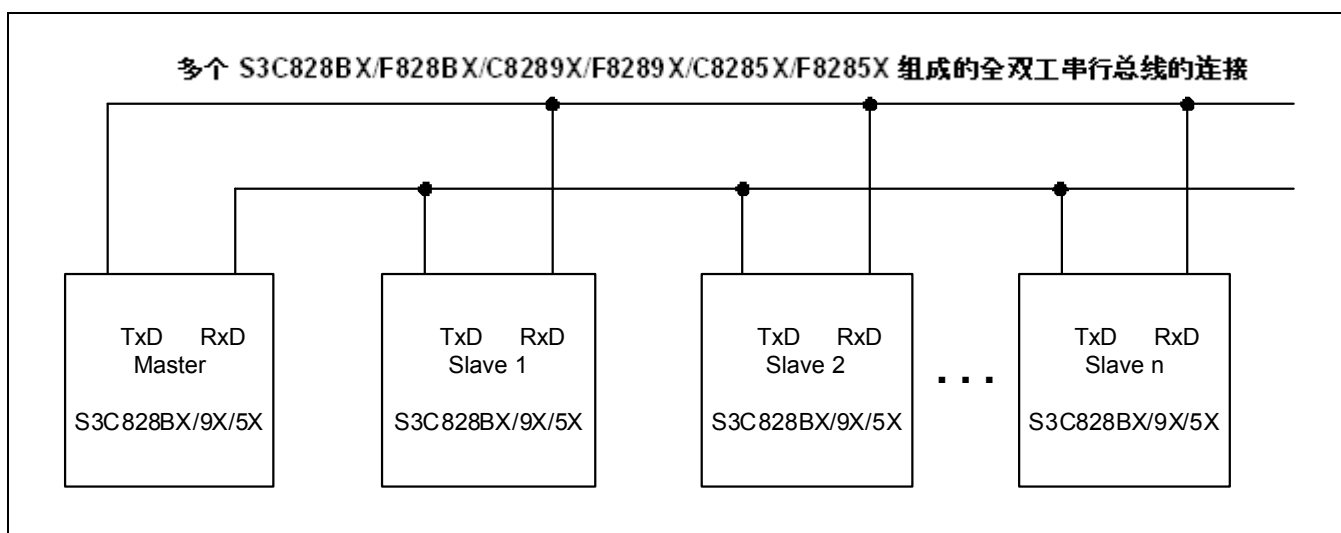


图 17-10 多节点串行通讯连接示例

# 18

## 电池电压检测

### 18.1 概述

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X内建一个BLD（电池电压检测）电路，用于检测电源电压跌落或者软件方式检测外部输入电平。BLD电路的开关可由软件控制。因为BLD工作时的电流很大，所以建议非必要时将BLD保持关闭。BLD阈值电压可以由软件设置。阈值电压可以设置成下列3种之一：

2.2V, 2.4V 或 2.8V ( $V_{DD}$  参考电压), 或者外部输入电平（外部参考电压）

BLD仅当 $B_{LDCON.3}$  置‘1’时工作。如果 $V_{DD}$  电平低于由 $B_{LDCON.2-0}$ 位选择的参考电压时， $B_{LDCON.4}$  将被置‘1’。如果 $V_{DD}$  电平高于参考电压时， $B_{LDCON.4}$  将被清零。低功耗应用时，关闭BLD模块。

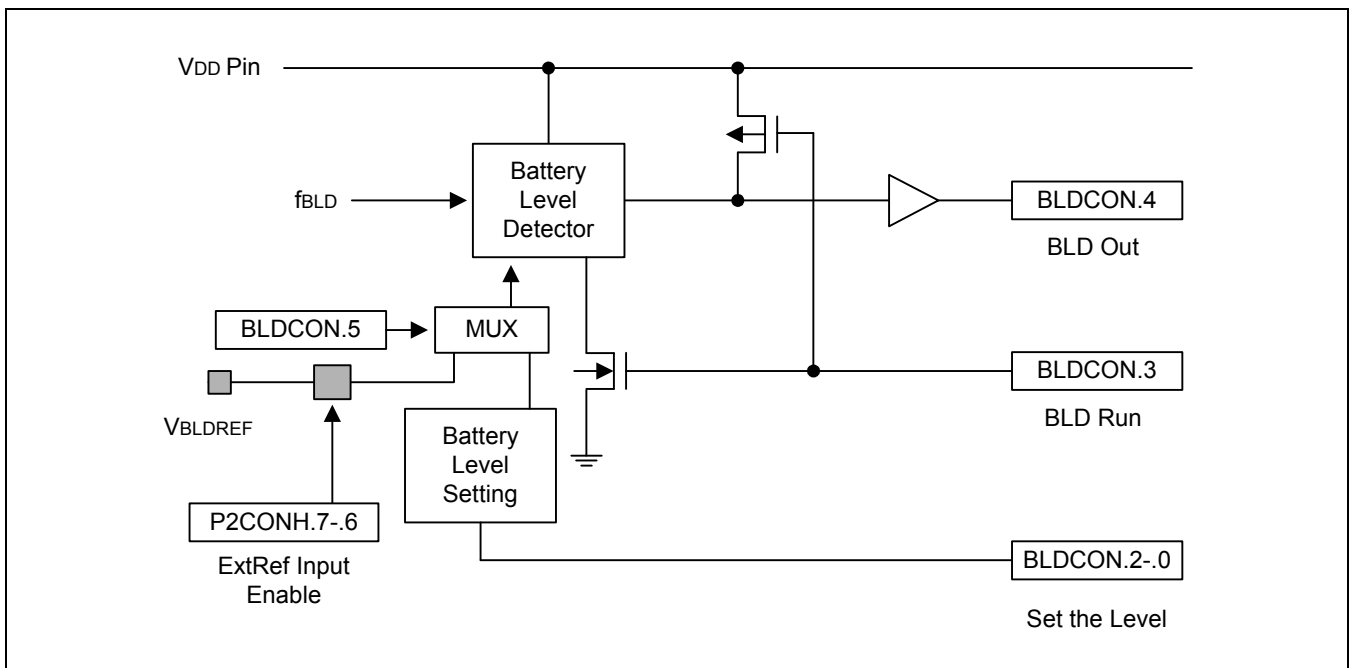


图 18-1 电池电压检测模块图

### 18.1.1 电池电压检测控制寄存器 (BLDCON)

BLDCON.3 位控制电池电压检测的开关。基本地，系统初始化后 $V_{BLD}$  的值为2.2 V，通过选择电池电压检测控制寄存器(BLDCON) 可将 $V_{BLD}$  设置成其他2个值。当向BLDCON中写入 3位数据时，内建的电阻串被选通并由此确定 $V_{BLD}$  的值。图 18-2 显示了3个不同的 $V_{BLD}$  值。

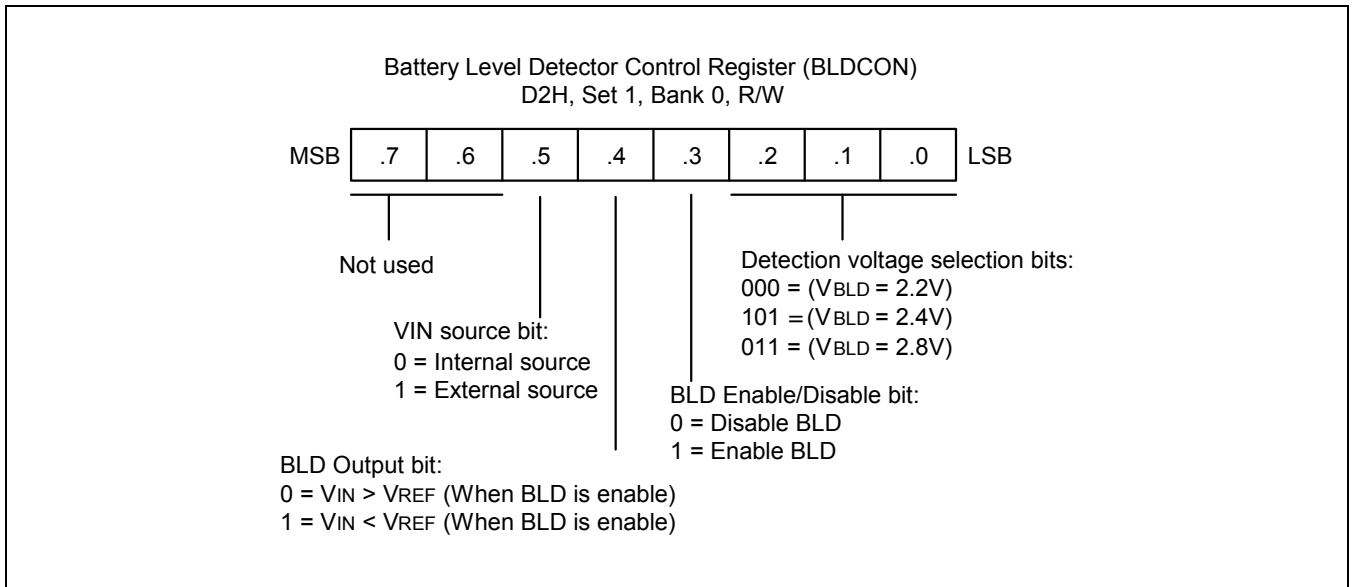


图 18-2 电池电压检测控制寄存器 (BLDCON)

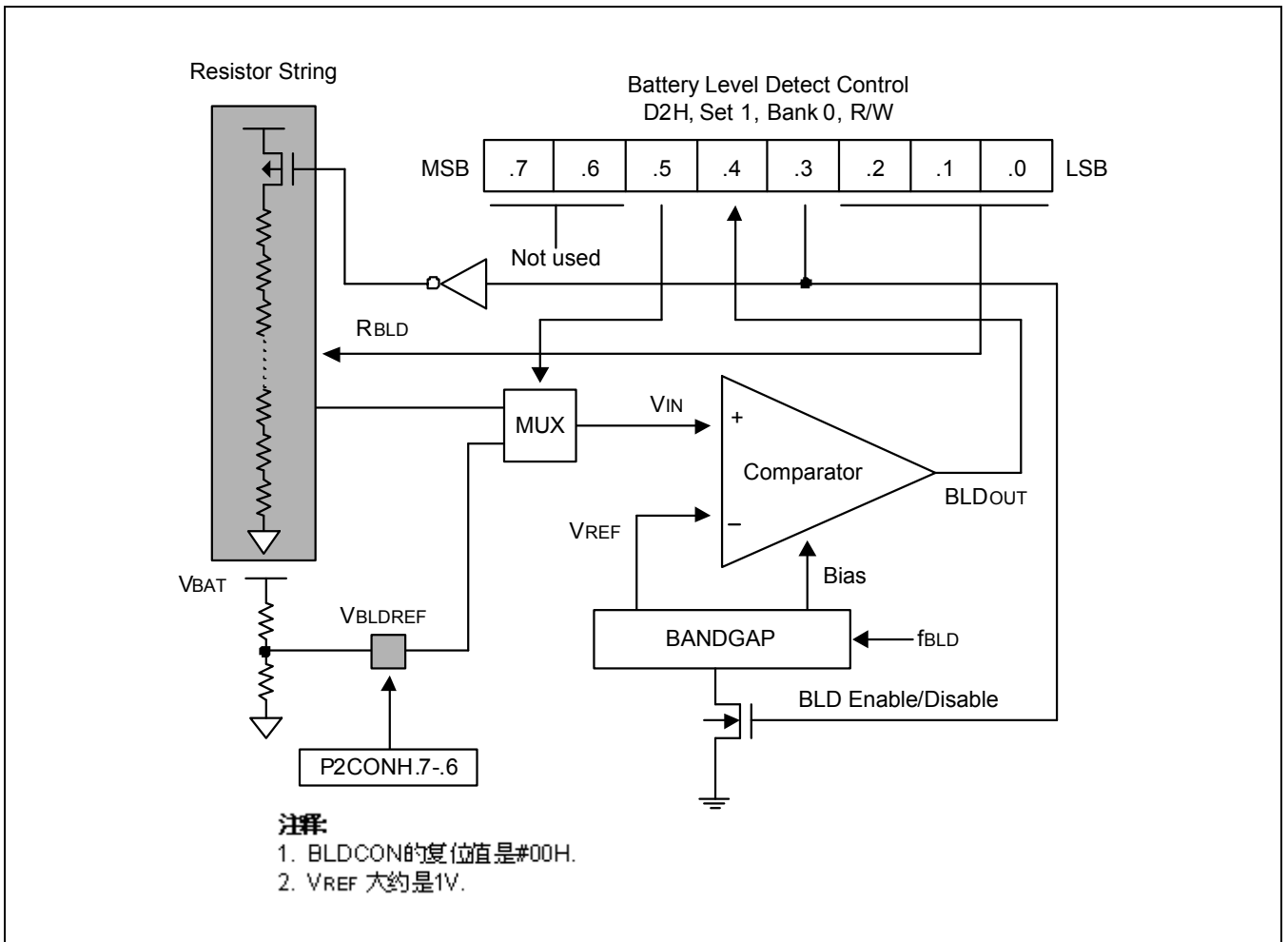


图 18-3 电池电压检测电路和模块图

表 18-1 BLDCON 值和检测电压

BLDCON .2-.0	V <sub>BLD</sub>
0 0 0	2.2V
1 0 1	2.4V
0 1 1	2.8V
其他	不可用



# 19

## 嵌入式闪存接口

### 19.1 概述

本章只针对S3F828BX，因为只有S3F828BX有片上Full Flash。该Flash支持'LDC'指令访问。编程的最小单位是字节，擦除的最小单位是扇区。用户可在任何时间往闪存里写数据。S3F828BX 64K字节闪存有两种工作模式：

- 用户编程模式：仅限于 S3F828BX
- 工具编程模式：参考第 22 章，S3F828BX/F8289X/F8285X FLASH MCU

## 19.2 用户编程模式

用户编程模式支持扇区擦除、字节编程、字节读取和一种保护模式（Hard Lock 保护）。

只有在工具编程模式下才可以启动读保护，所以用户需要在编程工具上勾选读保护项，使得编程完成后的芯片进入读保护状态。

S3F828BX 内部集成了自升压电路，可以产生高电压。因此，不需要在  $V_{PP}$  (Test) 管脚另加 12.5V 的编程电压，只是利用芯片内部逻辑电平就可以完成对闪存的操作。用户编程模式下的片上闪存编程是通过几个专用控制寄存器来实现的。

### 19.2.1 闪存控制寄存器（用户编程模式）

闪存控制寄存器 (FMCON)

FMCON 寄存器只支持用户编程模式下选择闪存操作模式：扇区擦除、字节编程、和设置硬件锁。

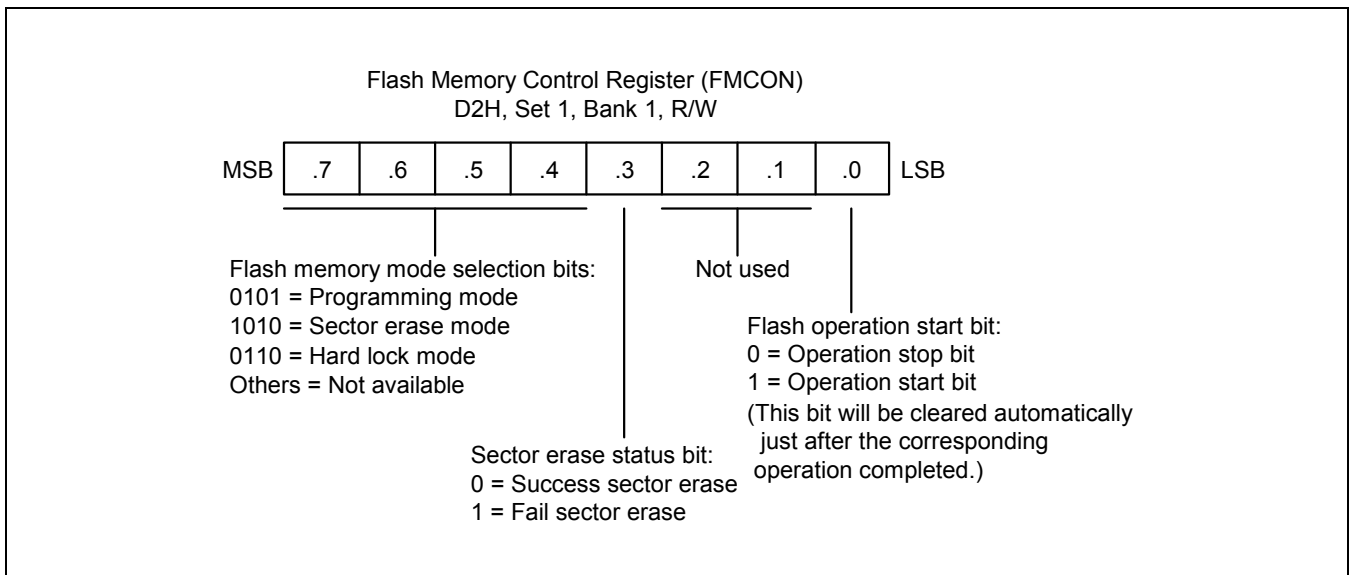


图 19-1 Flash Memory Control Register (FMCON)

FMCON.0 是擦除和 Hard Lock 保护操作的开始位。因此，设置 FMCON.0 为“1”，则开始擦除或 Hard Lock 保护操作。在执行扇区擦除或 Hard Lock 后，需要等待一段时间才可以用‘LDC’指令读或者写同一扇区的单元。闪存的读写则不需要用到这一控制位。

扇区擦除状态位是只读的。在执行扇区擦除的过程中，一旦发出中断请求，扇区擦除操作会被停止，CPU 响应中断请求。所以，为了操作的可靠性，必须在擦除操作结束后检查扇区擦除状态位，如果为‘0’则表示操作成功，为‘1’则表示操作失败，需要重新擦除。

**注释：** 当向 FMUSR 中写入“A5H”后，FMCON 的误操作或者程序跑飞都可能启动闪存擦除，用户编程或 Hard Lock 保护。需要小心避免上述情况的发生。

### 19.2.2 闪存用户编程使能寄存器 (FMUSR)

FMUSR 寄存器是为闪存的安全操作设置的。当 CPU 由于电噪声或其它原因致使程序跑飞时，这个寄存器的状态可以避免闪存的误操作。

复位后，由于 FMUSR 的值为“00000000B”，用户编程模式处于禁止状态。如需操作闪存，可通过设置 FMUSR 的值为“10100101B”来使能用户编程模式。除“10100101B”外的其它值，将禁止用户编程模式。

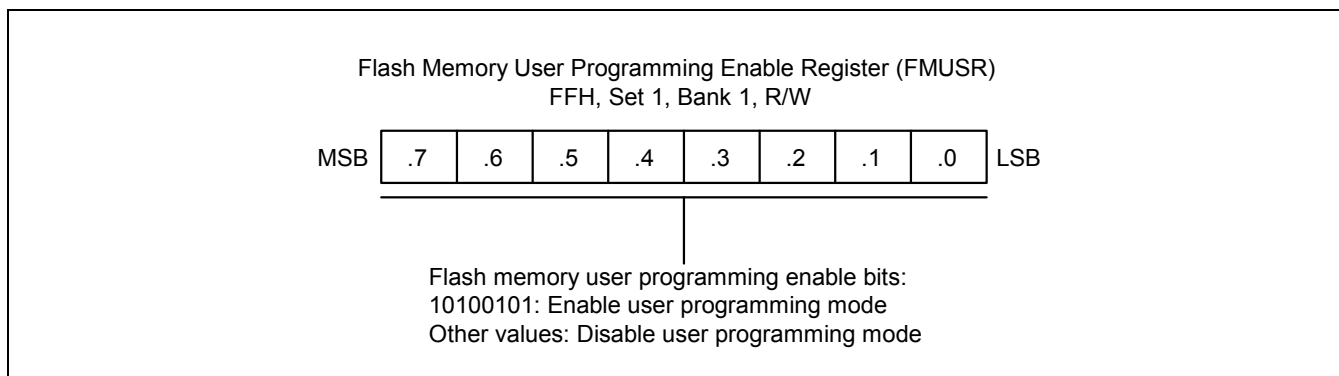


图 19-2 闪存用户编程使能寄存器 (FMUSR)

### 19.2.3 闪存扇区地址寄存器

擦除或编程闪存需要两个8位扇区地址寄存器来指定操作地址。FMSECL（低字节闪存扇区地址寄存器）存储被操作扇区基址的低8位地址，FMSECH（高字节闪存扇区地址寄存器）存储高8位地址。

S3F828BX 8里有512个扇区，所以需要用到FMSECH。一个扇区由128个字节组成。每个扇区的起始地址不是XX00H就是XX80H。所以FMSECL的低7位数据没有意义。但从操作的简单性来考虑，我们建议用户还是直接向FMSECH和FMSECL寄存器写入完整的扇区基址。

当编程闪存时，用户应该在确定扇区基址后再进行编程。如果下一个操作也是写入一字节数据，用户应该核对下一个目的地址是否位于同一个扇区内。如果不在同一扇区，用户应该重新向FMSECH和FMSECL寄存器中写入新的扇区基址。



图 19-3 闪存扇区地址寄存器高字节 (FMSECH)



图 19-4 闪存扇区地址寄存器低字节 (FMSECL)

### 19.3 ISP™ (在板编程) 扇区

ISP™ 扇区可以用于存放在板编程代码（通过 I/O 接口实现软件升级的启动代码）。

出于安全考虑，“LDC”指令擦除或编程操作不会影响 ISP™ 扇区的内容，但是这要通过使能 ISP 保护来实现。

用户可以通过设置 Smart Option 中 ISP 使能禁止位为“0”来使能芯片的 ISP 功能。如果不想使用 ISP 扇区，只要将 Smart Option 中 ISP 使能禁止位设为“1”，此时这个区域可以用作普通的程序存储空间。工具模式下，ISP 扇区会失去保护，即通过串行编程工具还是可以擦除或编程该区域。

ISP 扇区的大小可以通过 Smart Option 来设置。根据在板编程代码的大小选择合适的 ISP 扇区尺寸。

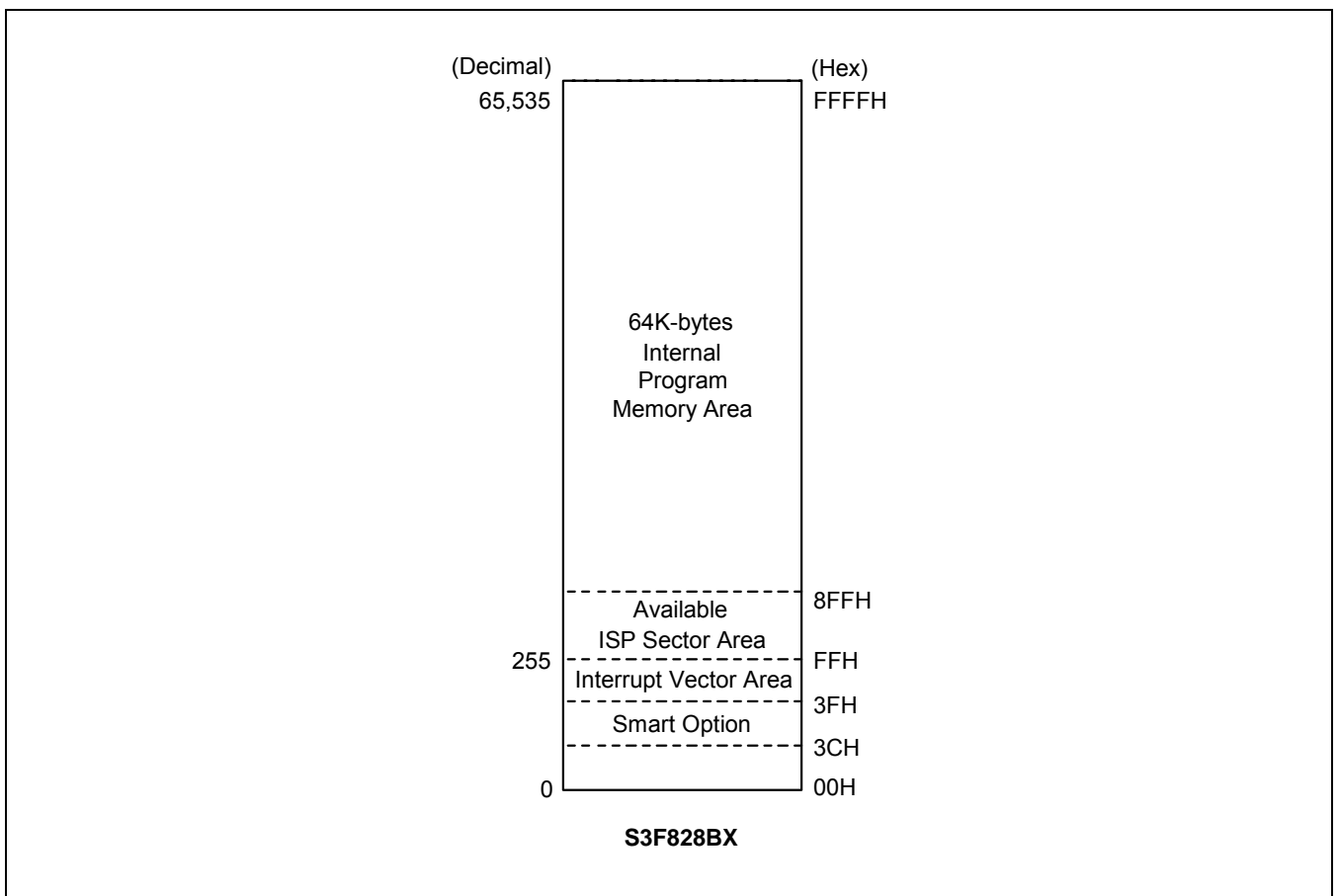


图 19-5 程序存储地址空间

表 19-1 ISP扇区大小

Smart Option(003CH) ISP扇区大小选择位			ISP 扇区	ISP 扇区大小
Bit 2	Bit 1	Bit 0		
1	x	x	–	0
0	0	0	100H – 1FFH ( 256 Byte)	256 Bytes
0	0	1	100H – 2FFH ( 512 Byte)	512 Bytes
0	1	0	100H – 4FFH (1024 Byte)	1024 Bytes
0	1	1	100H – 8FFH (2048 Byte)	2048 Bytes

**注释:** 在用户模式下，通过 Smart Option 位（3E.2-3E.0）设置的 ISP 扇区空间内，用 LDC 指令实现的擦除和编程操作都是无效的。

### 19.3.1 ISP复位向量和 ISP 扇区大小的关系

用户使能ISP 保护功能（3EH.2 = 0）并且选择了 OBP 复位向量（3EH.7 = 0）后，需要通过设置 ISP 复位向量地址选择位（3E.7-.5）来选择恰当的 CPU 复位地址（参考表 19-2）。

表 19-2 复位向量地址

Smart Option (003CH) ISP 复位向量地址选择			上电复位后的复位向 量地址	存放ISP代码的空间	ISP 扇区大小
Bit 7	Bit 6	Bit 5			
1	x	x	0100H	–	–
0	0	0	0200H	100H – 1FFH	256 Bytes
0	0	1	0300H	100H – 2FFH	512 Bytes
0	1	0	0500H	100H – 4FFH	1024 Bytes
0	1	1	0900H	100H – 8FFH	2048 Bytes

**注释:** Smart Option 中 ISP 复位向量选择位（003EH.7 – 003EH.5）和 ISP 保护扇区选择（003EH.2 – 003EH.0）是相互独立的。所以设置的时候要注意一致性。

### 19.3.2 扇区擦除

用户只能在用户编程模式下利用扇区擦除操作来部分擦除闪存。在用户编程模式下，擦除闪存的唯一单元是扇区。

S3F828BX 的 64K 字节闪存空间分为 512 个扇区。每个扇区大小为 128 个字节。因此，在对闪存内单个或多个字节进行编程前，必须先擦除目的地址所在的扇区。

在设置完被擦扇区的扇区地址并置高擦除起始位（FMCON.0）开始擦除后，至少需要 10 ms 来完成擦除操作。工具编程模式不支持扇区擦除（MDS 模式工具或编程工具）。

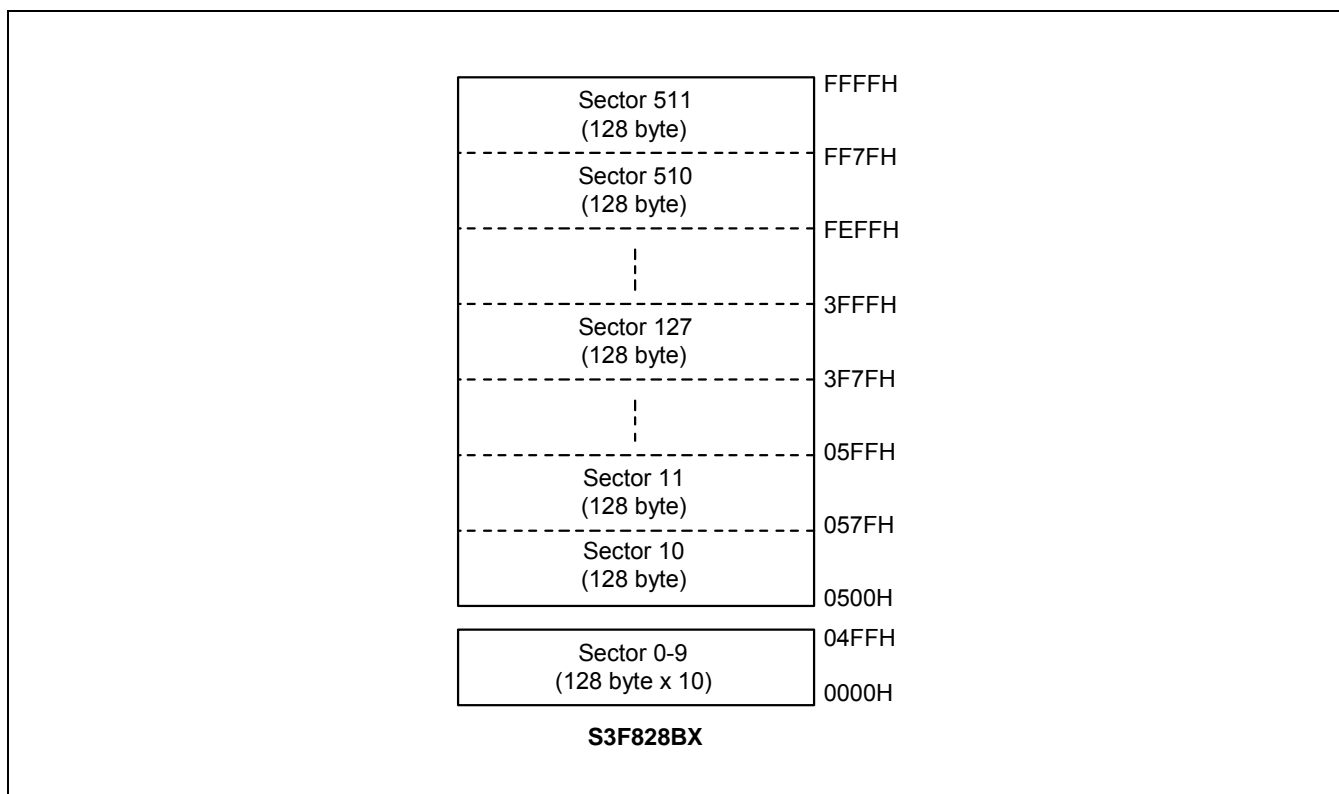


图 19-6 用户编程模式下的扇区

### 19.3.3 用户编程模式下扇区擦除流程

1. 设置闪存用户编程使能寄存器 (FMUSR) 为“10100101B”。
2. 设置闪存扇区地址寄存器 (FMSECH 和 FMSECL)。
3. 检查用户ID (用户自设置)
4. 设置闪存控制寄存器 (FMCON) 为“10100001B”。
5. 设置闪存用户编程使能寄存器 (FMUSR) 为“00000000B”
6. 检查“扇区擦除状态位” 判断此次擦除操作是否成功。

#### 编程实例 19-1 扇区擦除

```

      •
      •
reErase: SB1
          LD    FMUSR, #0A5H           ; 使用用户编程模式
          LD    FMSECH, #10H
          LD    FMSECL, #00H          ; 设置扇区地址 (1000H-107FH)
          CP    UserID_Code, #User_value ; 检查用户ID (用户自设置)
                                          ; User_value 可以是用户自设的任意值
          JR    NE, Not_ID_Code        ; 如果不相等, 则跳转到 Not_ID_Code
          LD    FMCON, #10100001B     ; 开始扇区擦除操作
          NOP                                     ; 必需的空指令
          NOP                                     ; 必需的空指令
          LD    FMUSR, #0              ; 关闭用户编程模式
          TM    FMCON, #00001000B     ; 检查 “扇区擦除状态位”
          JR    NZ, reErase            ; 如果擦除失败则跳转到reErase
      •
      •
      •
      •
Not_ID_Code:
          SB1
          LD    FMUSR, #0              ; 关闭用户编程模式
      •
      •
      •
      •

```



### 19.3.4 编程

扇区擦除后，闪存编程操作是以一个字节为单位来进行的。为了Flash内容的安全性，必须设置 FMSECH 和 FMSECL 为闪存中欲操作区间的扇区地址。

由“LDC”指令开始编程写操作，在改变 FMSECH 和 FMSECL 之前最多可以连续写入128 字节，因为一个扇区的大小就是128 字节。

#### 19.3.4.1 用户编程模式下的编程流程

- 编程前必须保证编程地址中的值为“FFH”（可以通过扇区擦除来实现）。
- 设置闪存用户编程使能寄存器 (FMUSR) 为“10100101B”。
- 设置闪存扇区地址寄存器 (FMSECH 和 FMSECL) 为要写数据的扇区基地址的值
- 把闪存的高位地址放进对工作寄存器的高地址寄存器中。
- 把闪存的低位地址放进对工作寄存器的低地址寄存器中。
- 把要发送的数据放到一个工作寄存器中。
- 检查用户ID (用户自设)
- 设置闪存控制寄存器 (FMCON) 为“01010001B”。
- 用“LDC”指令，通过间接寻址方式把要发送的数据写入闪存地址区。
- 设置闪存用户编程使能寄存器 (FMUSR) 为“00000000B”，以关闭用户编程模式。

## 编程实例 19-2 编程

```

•
•

SB1
LD    FMUSR, #0A5H           ; 使用户编程模式
LD    FMSECH, #17H
LD    FMSECL, #80H         ; 设置扇区地址 (1780H-17FFH)
LD    R2, #17H             ; 设置位于同一扇区的目标操作地址
LD    R3, #84H
LD    R4, #78H             ; 临时数据
CP    UserID_Code, #User_value ; 检查用户ID (用户自设)
                                   ; User_value可以是用户设置的任意值
JR    NE, Not_ID_Code      ; 如果不相等, 则跳转 Not_ID_Code
LD    FMCON, #01010000B    ; 开始编程
LDC   @RR2, R4             ; 往1784H单元写数据
NOP                                       ; 必需的空指令
LD    FMUSR, #0            ; 关闭用户编程模式

•
•
•
•

Not_ID_Code:
SB1
LD    FMUSR, #0            ; 关闭用户编程模式

•
•
•
•

```

### 19.3.5 读

由“LDC”指令开始读操作。

用户编程模式下编程流程

1. 把闪存的高位地址放进对工作寄存器的高地址寄存器中
2. 把闪存的低位地址放进对工作寄存器的低地址寄存器中
3. 用“LDC”指令，通过间接寻址方式从闪存区读取数据。

#### 编程实例 19-3 读

```

      •
      •

      LD    R2, #3H      ; 把闪存的高位地址放进工作寄存器对的高地址寄存器中
      LD    R3, #0       ; 低位地址放进工作寄存器对的低地址寄存器中
LOOP:  LDC   R0, @RR2    ; 从闪存区读数据
                               ; (在 300H 和 3FFH之间)

      INC   R3
      CP    R3, #0H
      JP    NZ, LOOP

      •
      •
      •
      •

```

## 19.4 HARD LOCK 保护

用户可通过写 ‘0110B’ 到 FMCON7-4 开启 Hard Lock 保护。这个功能可以防止闪存区数据的变化。如果启用这个功能，用户就不能往闪存里写或擦除数据。可通过在工具编程模式下，执行片擦除操作来接触这种保护。

在用户编程模式下，设置 Hard Lock 保护的流程如下。工具模式下，工具的制造商通过编程选项支持硬件保护。具体请参照制造商提供的编程工具用户说明书。

用户编程模式下的编程流程

1. 设置闪存用户编程使能寄存器 (FMUSR) 为“10100101B”。
2. 检查设置闪存控制寄存器 (FMCON) 为“01100001B”。
3. 检查用户ID (用户自设)
4. 设置闪存用户编程使能寄存器 (FMUSR) 为“00000000B”。

### 编程实例 19-4 Hard Lock 保护

```

•
•
SB1
LD    FMUSR,#0A5H           ; 使能用户编程模式
CP    UserID_Code,#User_value ; 检查用户ID (用户自设)
                                ; User_value 可以是用户设定的任何值
JR    NE,Not_ID_Code       ; 如果不相等, 则跳转 Not_ID_Code
LD    FMCON,#01100001B     ; 设置并开始Hard Lock
NOP                                       ; 必需的空指令
LD    FMUSR,#0             ; 关闭用户编程模式
•
•
•
•
Not_ID_Code:
SB1
LD    FMUSR,#0             ; 关闭用户编程模式
•
•
•
•

```

# 20

## 电气参数

### 20.1 概述

本章将以表格或图表的方式提供 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X的电气参数。按如下顺序排列：

- 芯片极限物理特性
- 输入输出电容
- 直流电气特性
- 交流电气特性
- 振荡器特性
- 振荡器稳定时间
- STOP 模式下，RAM数据保持电压
- LVR 复位时序
- BLD 电气特性
- SIO 时序
- A/D 转换电气特性
- UART 时序
- 内部 Flash ROM 电气特性
- 工作电压范围

表 20-1 芯片极限物理特性

 $(T_A = 25^\circ\text{C})$ 

参数	标号	条件	范围	单位
供电电压	$V_{DD}$	—	$-0.3 - +4.6$	V
输入电压	$V_I$	P0-8 口	$-0.3 - V_{DD} + 0.3$	V
输出电压	$V_O$	—	$-0.3 - V_{DD} + 0.3$	
I/O 口输出电流	$I_{OH}$	单个 I/O 口工作	- 15	
		所有 I/O 口工作	- 60	
I/O 口输入电流	$I_{OL}$	单个 I/O 口工作	+ 30	mA
		所有的 I/O 口的电流之和	+ 100	
工作温度	$T_A$	—	$-25 - +85$	$^\circ\text{C}$
储藏温度	$T_{STG}$	—	$-65 - +150$	

表 20-2 直流电气特性

 $(T_A = -25^\circ\text{C} - +85^\circ\text{C}, V_{DD} = 2.0\text{V} - 3.6\text{V})$ 

参数	标号	条件	最小值	典型值	最大值	单位
工作电压	$V_{DD}$	$f_x = 0.4-4.2\text{MHz}$ , $f_{xt} = 32.768\text{kHz}$	2.0	—	3.6	V
		$f_x = 0.4-10\text{MHz}$	2.7	—	3.6	
		$f_x = 0.4-11.1\text{MHz}$	3.0	—	3.6	
输入高电平	$V_{IH1}$	除 $V_{IH2}$ , $V_{IH3}$ 以外所有输入管脚	$0.7V_{DD}$	—	$V_{DD}$	V
	$V_{IH2}$	P0-1 口, nRESET	$0.8V_{DD}$	—	$V_{DD}$	
	$V_{IH3}$	$X_{IN}$ , $X_{OUT}$ 和 $XT_{IN}$ , $XT_{OUT}$	$V_{DD}-0.1$	—	$V_{DD}$	
输入低电平	$V_{IL1}$	除 $V_{IL2}$ , $V_{IL3}$ 以外所有输入管脚	—	—	$0.3V_{DD}$	V
	$V_{IL2}$	P0-1 口, nRESET	—	—	$0.2V_{DD}$	
	$V_{IL3}$	$X_{IN}$ , $X_{OUT}$ 和 $XT_{IN}$ , $XT_{OUT}$	—	—	0.1	
输出高电平	$V_{OH}$	$V_{DD} = 2.7\text{V} - 3.6\text{V}$ $I_{OH} = -1\text{mA}$ 所有输出管脚	$V_{DD}-1.0$	—	—	V
输出低电平	$V_{OL1}$	$V_{DD} = 2.7\text{V} - 3.6\text{V}$ $I_{OL} = 15\text{mA}$ P1-2 口	—	—	1.0	V
	$V_{OL2}$	$V_{DD} = 2.7\text{V} - 3.6\text{V}$ $I_{OL} = 10\text{mA}$ 除 $V_{OL1}$ 以外所有输出管脚	—	—	1.0	

参数	标号	条件	最小值	典型值	最大值	单位	
输入高电平漏电流	$I_{LIH1}$	$V_{IN} = V_{DD}$ 除 $I_{LIH2}$ 以外所有输入管脚	–	–	3	$\mu A$	
	$I_{LIH2}$	$V_{IN} = V_{DD}$ $X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}$			20		
输入低电平漏电流	$I_{LIL1}$	$V_{IN} = 0V$ 除nRESET, $I_{LIL2}$ 以外所有输入管脚	–	–	–3		
	$I_{LIL2}$	$V_{IN} = 0V$ $X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}$			–20		
输出高电平漏电流	$I_{LOH}$	$V_{OUT} = V_{DD}$ 所有输出管脚	–	–	3		
输出低电平漏电流	$I_{LOL}$	$V_{OUT} = 0V$ 所有输出管脚	–	–	–3		
LCD 分压电阻	$R_{LCD}$	$T_A = 25^\circ C$	25	50	80	$k\Omega$	
振荡器反馈电阻	$R_{OSC1}$	$V_{DD} = 3V, T_A = 25^\circ C$ $X_{IN} = V_{DD}, X_{OUT} = 0V$	600	1600	3000		
	$R_{OSC2}$	$V_{DD} = 3V, T_A = 25^\circ C$ $XT_{IN} = V_{DD}, XT_{OUT} = 0V$	2000	4000	8000		
上拉电阻	$R_{L1}$	$V_{IN} = 0V; V_{DD} = 3V$ P0–8 口, $T_A = 25^\circ C$	40	70	100		
	$R_{L2}$	$V_{IN} = 0V; V_{DD} = 3V$ $T_A = 25^\circ C, nRESET$	220	360	500		
中间输出电压 (1)	$V_{LC1}$	$V_{DD} = 2.7V$ 到 $3.6V, 1/4$ 偏置 LCD 时钟 = 0Hz, $V_{LC0} = V_{DD}$	$0.75V_{DD} - 0.2$	$0.75V_{DD}$	$0.75V_{DD} + 0.2$	V	
	$V_{LC2}$		$0.5V_{DD} - 0.2$	$0.5V_{DD}$	$0.5V_{DD} + 0.2$		
	$V_{LC3}$		$0.25V_{DD} - 0.2$	$0.25V_{DD}$	$0.25V_{DD} + 0.2$		
$ V_{LCD} - COM_i $ 压降( $i = 0 - 7$ )	$V_{DC}$	每个Common 管脚 – $15\mu A$	–	–	120	mV	
$ V_{LCD} - SEG_x $ 压降( $x = 0 - 34$ )	$V_{DS}$	每个Segment 管脚 – $15\mu A$	–	–	120		
供电电流(2)	$I_{DD1}^{(3)}$	Run 模式: $V_{DD} = 3.3V \pm 0.3V$ 石英晶体振荡器 $C1 = C2 = 22pF$	11.1MHz	–	4.0	8.0	mA
			4.0MHz	–	1.8	3.6	
	$I_{DD2}^{(3)}$		11.1MHz	–	1.0	2.0	
			4.0MHz		0.5	1.0	

参数	标号	条件	最小值	典型值	最大值	单位
		C1 = C2 = 22pF				
	$I_{DD3}^{(4)}$	Run模式: $V_{DD} = 3.3V \pm 0.3V$ , $T_A = 25^\circ C$ , OSCON.7 = 1 32kHz石英晶体振荡器	–	14.0	28.0	$\mu A$
	$I_{DD4}^{(4)}$	Idle模式: $V_{DD} = 3.3V \pm 0.3V$ , $T_A = 25^\circ C$ , OSCON.7 = 1 32kHz石英晶体振荡器	–	2.0	4.0	
	$I_{DD5}^{(5)}$	Stop 模式: $V_{DD} = 3.3V \pm 0.3V$	$T_A = 25^\circ C$	–	0.2	2.0
$T_A = -25^\circ C$ – $85^\circ C$			–	–	10	

**注释:**

1. 当  $V_{LC0}$  开路时的中间输出电压
2. 供电电流不包括内部上拉电阻, LCD 分压电阻, LVR 模块消耗电流以及外部输出电流负载。
3.  $I_{DD1}$  和  $I_{DD2}$  包含副时钟振荡的功耗。
4.  $I_{DD3}$  和  $I_{DD4}$  是主时钟振荡停止, 副时钟使用 (OSCCON.7 = 1) 时的电流。
5.  $I_{DD5}$  是主时钟和副时钟振荡都停止时的电流。
6. 表中所有值都是在系统时钟控制寄存器 (CLKCON.4–3) 的位 4-3 设置成 11B 时的结果。



表 20-3 交流电气特性

 $(T_A = -25^{\circ}\text{C} - +85^{\circ}\text{C}, V_{DD} = 2.0\text{V} - 3.6\text{V})$ 

参数	标号	条件	最小值	典型值	最大值	单位
中断输入高、低电平宽度(P0.0-P0.7)	$t_{INTL}, t_{INTH}$	所有外部中断, $V_{DD} = 3\text{V}$	500	700	-	ns
nRESET 输入低电平宽度	$t_{RSL}$	$V_{DD} = 3\text{V}$	10	-	-	$\mu\text{s}$

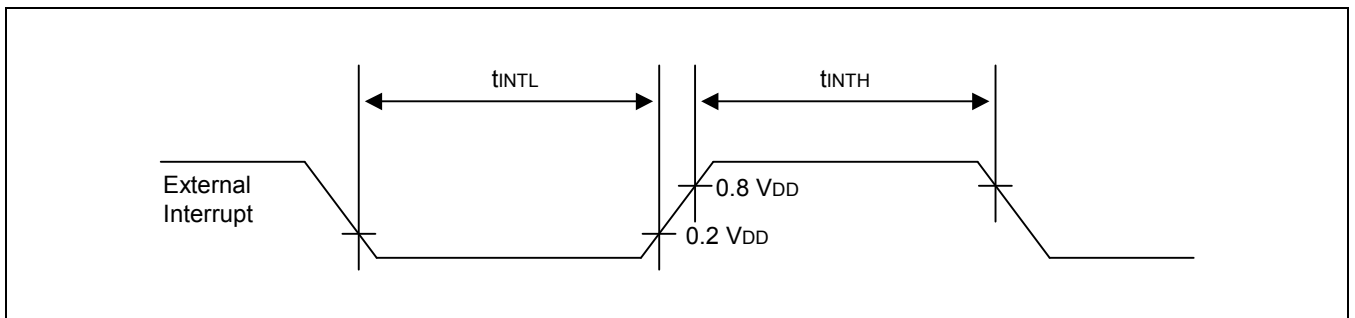


图 20-1 外部中断时序

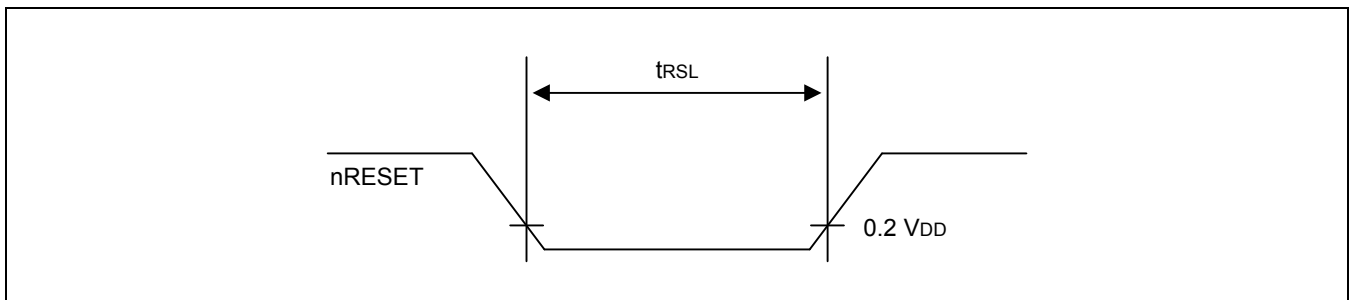


图 20-2 nRESET 输入时序

表 20-4 输入输出电容

 $(T_A = -25^{\circ}\text{C} - +85^{\circ}\text{C}, V_{DD} = 0\text{V})$ 

参数	标号	条件	最小值	典型值	最大值	单位
输入电容	$C_{IN}$	$f = 1\text{MHz}$ ; 不测试的管脚接 $V_{SS}$	-	-	10	pF
输出电容	$C_{OUT}$					
I/O 口电容	$C_{IO}$					

表 20-5 模式下数据保持电压

 $(T_A = -25^{\circ}\text{C} - +85^{\circ}\text{C})$ 

参数	标号	条件	最小值	典型值	最大值	单位
数据保持所需的供电电压	$V_{DDDR}$		2.0	-	3.6	V
数据保持所需的供电电流	$I_{DDDR}$	$V_{DDDR} = 2\text{V}$ Stop 模式, $T_A = 25^{\circ}\text{C}$ 禁止LVR 模块	-	-	1	$\mu\text{A}$

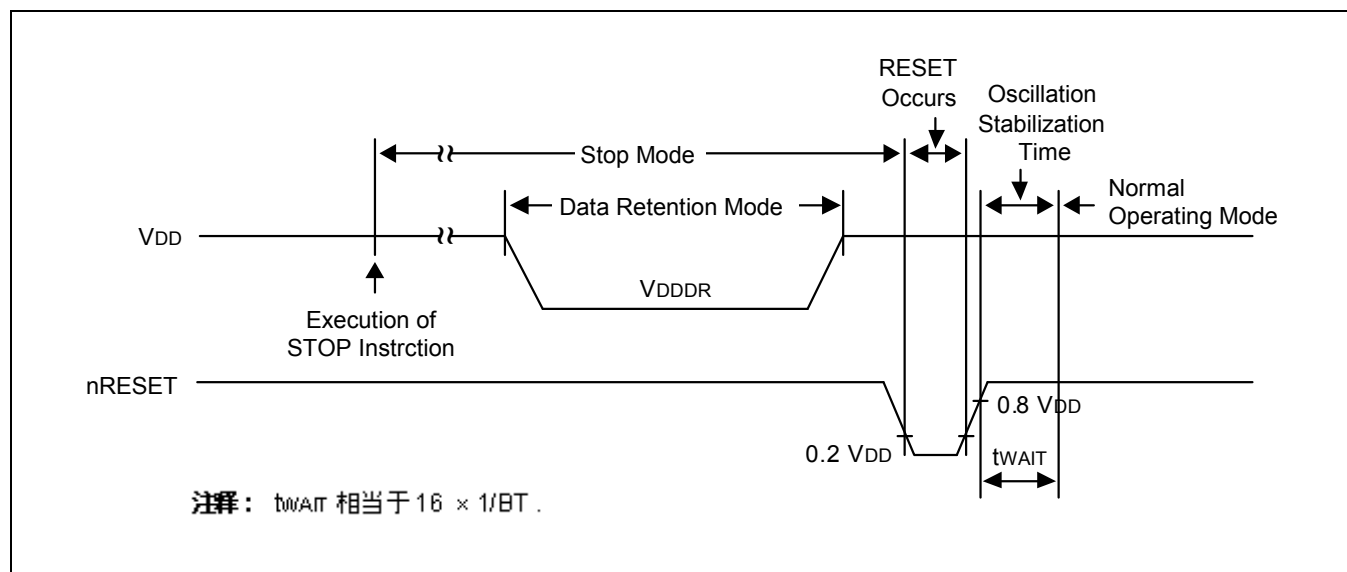


图 20-3 Stop 模式下复位时序

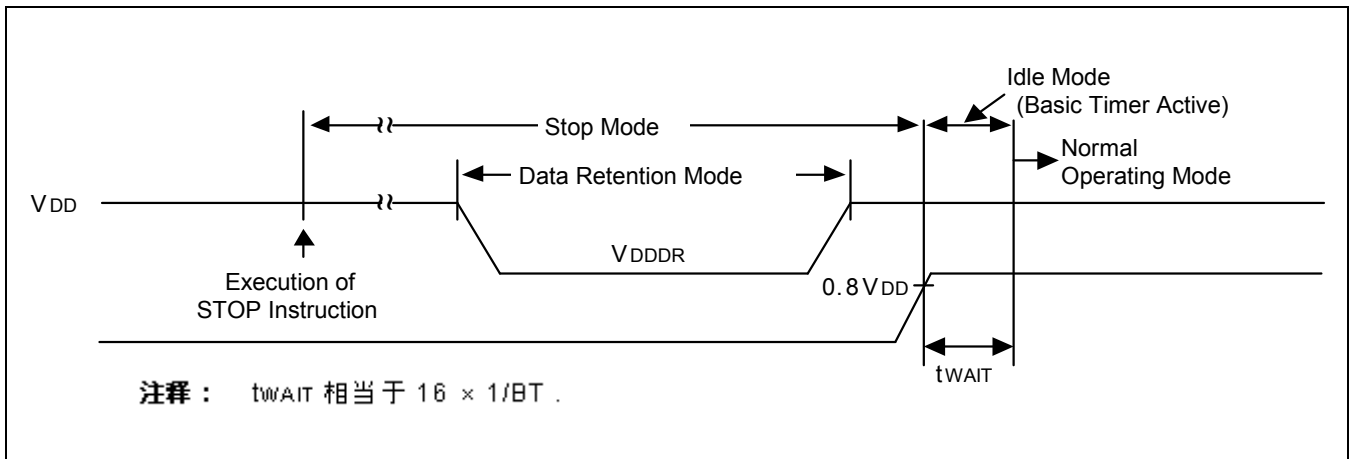


图 20-4 外部中断唤醒Stop 模式时序

表 20-6 A/D 转换电气特性

(T<sub>A</sub> = -25°C – + 85°C, V<sub>DD</sub> = 2.7V – 3.6V, V<sub>SS</sub> = 0V)

参数	标号	条件	最小值	典型值	最大值	单位
精度		–	–	10	–	bit
总误差		–	–	–	±3	LSB
积分线性误差	ILE	V <sub>DD</sub> = 3.072V V <sub>SS</sub> = 0V CPU 时钟 = 11.1MHz	–	–	±2	
微分线性误差	DLE		–	–	±1	
顶部偏移误差	EOT		±1	–	±3	
底部偏移误差	EOB		±1	–	±3	
转换时间 (1)	T <sub>CON</sub>	10位精度 50 x f <sub>xx</sub> /4, f <sub>xx</sub> = 8MHz	25	–	–	μS
模拟输入电压	V <sub>IAN</sub>	–	V <sub>SS</sub>	–	AV <sub>REF</sub>	V
模拟输入阻抗	R <sub>AN</sub>	–	2	1000	–	M
模拟参考电压	AV <sub>REF</sub>	–	2.0	–	V <sub>DD</sub>	V
模拟地	AV <sub>SS</sub>	–	V <sub>SS</sub>	–	V <sub>SS</sub> + 0.3	
模拟输入电流	I <sub>ADIN</sub>	V <sub>DD</sub> = 3.3V	–	–	5	μA
模拟模块电流 (2)	I <sub>ADC</sub>	V <sub>DD</sub> = 3.3V	–	0.5	1.5	mA
		V <sub>DD</sub> = 3.3V 省电模式		100	500	nA

**注释:**

- “转换时间”是启动转换到转换结束所需时间。
- I<sub>ADC</sub> 是 A/D 转换的工作电流。

表 20-7 低电压复位电气特性

 $(T_A = 25^\circ\text{C})$ 

参数	标号	条件	最小值	典型值	最大值	单位
LVR 电压	$V_{LVR}$	$T_A = 25^\circ\text{C}$	2.0	2.2	2.4	V
VDD 电压上升时间	$t_R$	—	10	—	—	$\mu\text{s}$
VDD 电压关断时间	$t_{OFF}$	—	0.5	—	—	S
LVR 时滞电压	$\Delta V$	—	—	10	100	mV
电流消耗	$I_{DDPR}$	$V_{DD} = 3.3\text{V}$	—	70	120	$\mu\text{A}$

注释： 在“Smart Option”中使能LVR后，LVR电路就有电流消耗。

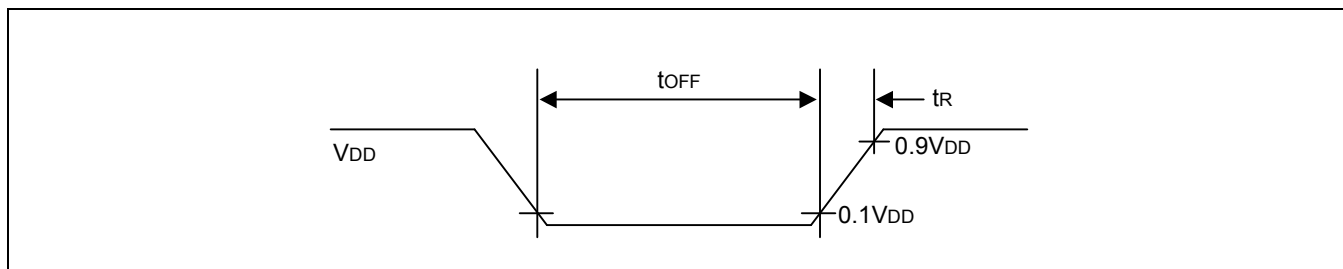


图 20-5 LVR (低电压复位) 时序

表 20-8 电池电压检测电气特性

 $(T_A = 25^\circ\text{C}, V_{DD} = 2.0\text{V} - 3.6\text{V})$ 

参数	标号	条件	最小值	典型值	最大值	单位
BLD工作电压	$V_{DDBLD}$	—	2.0	—	3.6	V
BLD电压	$V_{BLD}$	BLDCON.2-.0 = 000b	2.0	2.2	2.4	V
		BLDCON.2-.0 = 101b	2.15	2.4	2.65	
		BLDCON.2-.0 = 011b	2.5	2.8	3.1	
电流消耗	$\Delta V$	BLDCON.2-.0 = 000, 101, 011b	—	10	100	mV
BLD时滞电压	$I_{BLD}$	$V_{DD} = 3.3\text{V}$	—	70	120	$\mu\text{A}$
		$V_{DD} = 2.2\text{V}$	—	50	100	
BLD 电路响应时间	$T_B$	$f_w = 32.768\text{kHz}$	—	—	1	ms

表 20-9 同步 SIO 电气特性

 $(T_A = -25^{\circ}\text{C} - +85^{\circ}\text{C}, V_{DD} = 2.0\text{V} - 3.6\text{V})$ 

参数	标号	条件	最小值	典型值	最大值	单位
SCK 周期	$t_{KCY}$	外部 SCK 源	1,000	-	-	ns
		内部 SCK 源	1,000			
SCK 高、低电平宽度	$t_{KH}, t_{KL}$	外部 SCK 源	500			
		内部 SCK 源	$t_{KCY}/2-50$			
相对于SCK 高电平的SI建立时间	$t_{SIK}$	外部 SCK 源	250			
		内部 SCK 源	250			
相对于SCK 高电平的SI保持时间	$t_{KSI}$	外部 SCK 源	400			
		内部 SCK 源	400			
相对于SO的SCK 输出延时	$t_{KSO}$	外部 SCK 源	-	300		
		内部 SCK 源		250		

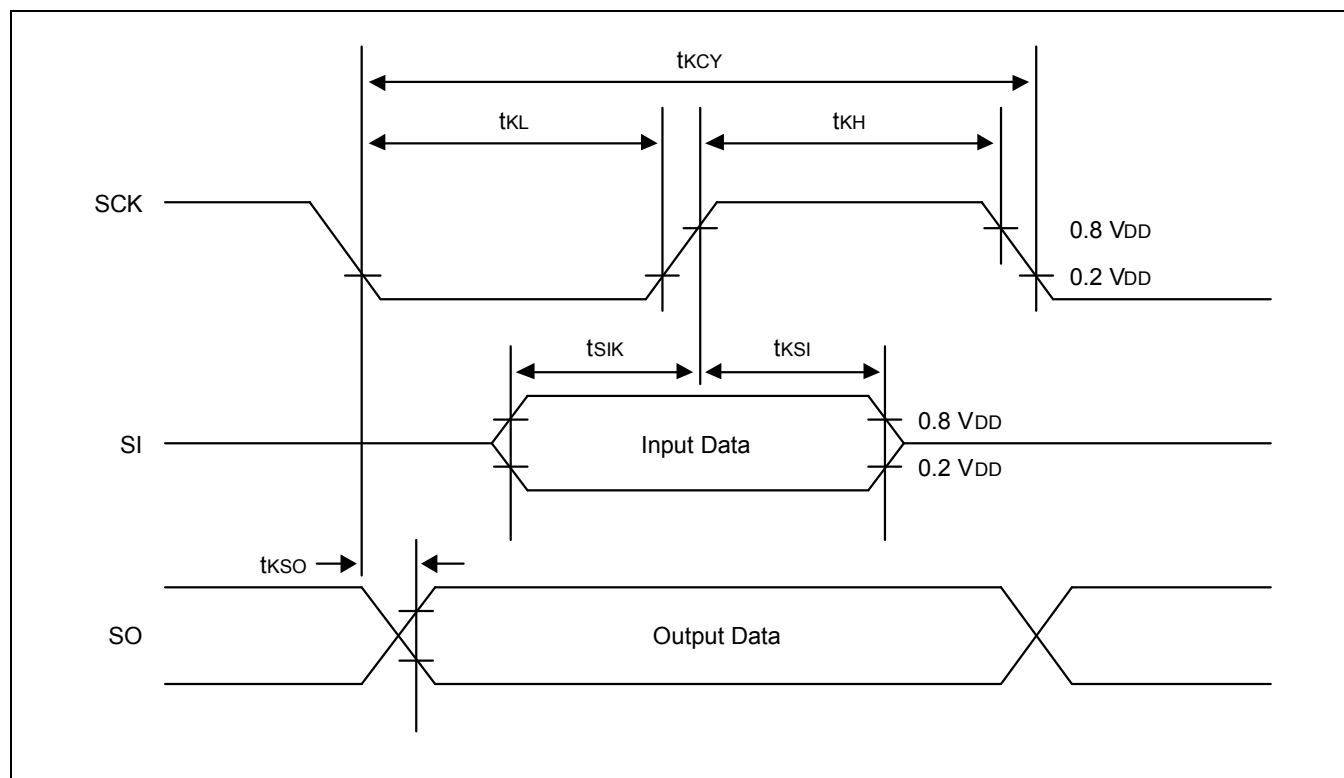


图 20-6 串行数据传输时序

表 20-10 UART Mode 0 时序特性(11.1MHz)

(T<sub>A</sub> = -25°C – + 85°C, V<sub>DD</sub> = 2.0V – 3.6V, Load Capacitance = 80pF)

参数	标号	条件	最小值	典型值	最大值
串行口时钟周期	t <sub>SCK</sub>	1,250	t <sub>CPU</sub> × 16	1,650	ns
相对于时钟上升沿的输出数据建立时间	t <sub>S1</sub>	590	t <sub>CPU</sub> × 13	–	
相对于输入数据有效的时钟上升沿	t <sub>S2</sub>	–	–	590	
时钟上升沿后的输出数据保持时间	t <sub>H1</sub>	t <sub>CPU</sub> – 50	t <sub>CPU</sub>	–	
时钟上升沿后的输入数据保持时间	t <sub>H2</sub>	0	–	–	
串行口时钟高、低电平宽度	t <sub>HIGH</sub> , t <sub>LOW</sub>	470	t <sub>CPU</sub> 8	970	

## 注释:

- 所有的时序都是纳秒 (ns) 级的, 并且 CPU 时钟为 11.1 MHz。
- 单位 t<sub>U</sub> 表示 1 个 UART 时钟周期。

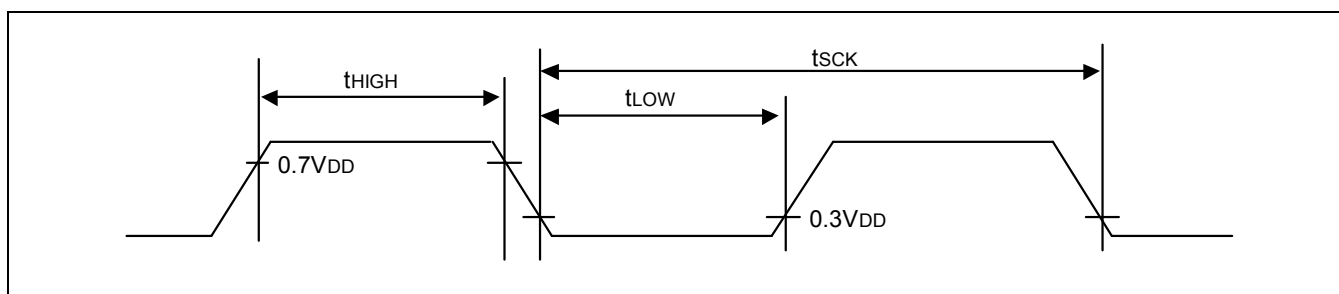


图 20-7 UART时序特性波形

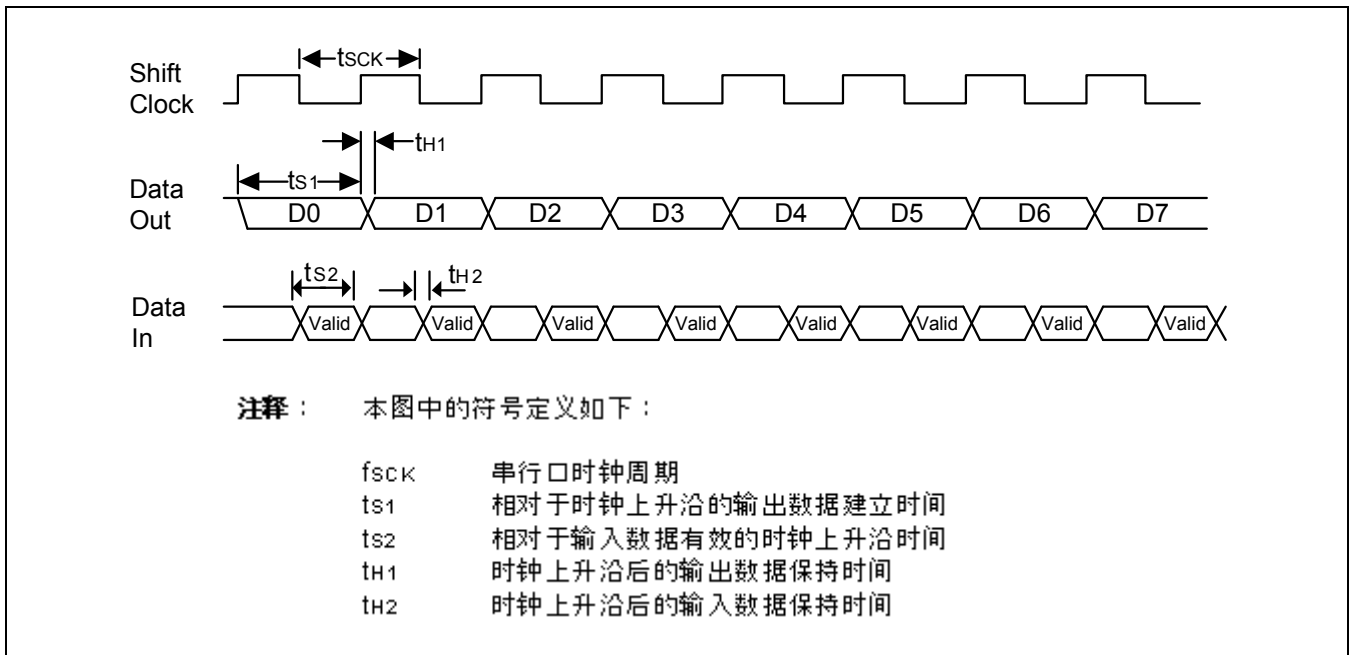


图 20-8 UART 模块的时序波形



表 20-11 主振荡器特性

 $(T_A = -25^{\circ}\text{C} - + 85^{\circ}\text{C})$ 

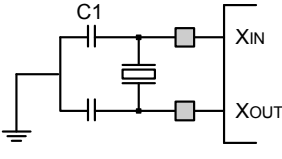
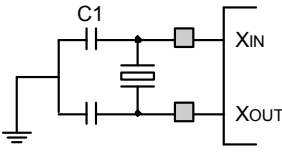
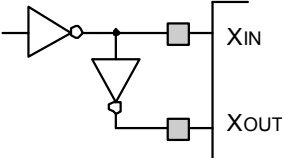
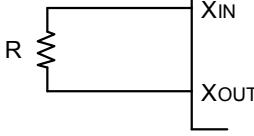
振荡器	时钟配置	参数	测试条件	最小值	典型值	最大值	单位
石英振荡器		主振荡频率	3.0V – 3.6V	0.4	–	11.1	MHz
			2.7V – 3.6V	0.4	–	10.0	
			2.0V – 3.6V	0.4	–	4.2	
陶瓷振荡器		主振荡频率	3.0V – 3.6V	0.4	–	11.1	MHz
			2.7V – 3.6V	0.4	–	10.0	
			2.0V – 3.6V	0.4	–	4.2	
外部时钟源		$X_{IN}$ 输入频率	3.0V – 3.6V	0.4	–	11.1	MHz
			2.7V – 3.6V	0.4	–	10.0	
			2.0V – 3.6V	0.4	–	4.2	
RC 振荡器		频率	3.3V	0.4	–	1.0	MHz

表 20-12 副振荡器特性

 $(T_A = -25^{\circ}\text{C} - + 85^{\circ}\text{C})$ 

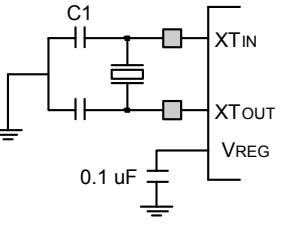
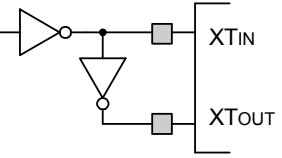
振荡器	时钟配置	参数	测试条件	最小值	典型值	最大值	单位
石英振荡器		副振荡频率	2.0V – 3.6V	32	32.768	35	kHz
外部时钟源		$XT_{IN}$ 输入频率	2.0V – 3.6V	32	–	100	

表 20-13 主振荡稳定时间

 $(T_A = -25^{\circ}\text{C} - +85^{\circ}\text{C}, V_{DD} = 2.0\text{V} - 3.6\text{V})$ 

振荡器	测试条件	最小值	典型值	最大值	单位
石英振荡器	$f_x > 1\text{MHz}$	—	—	40	ms
陶瓷振荡器	$V_{DD}$ 满足最小起振电压时振荡稳定。	—	—	10	ms
外部时钟源	$X_{IN}$ 输入高低电平宽度 ( $t_{XH}, t_{XL}$ )	62.5	—	1250	ns

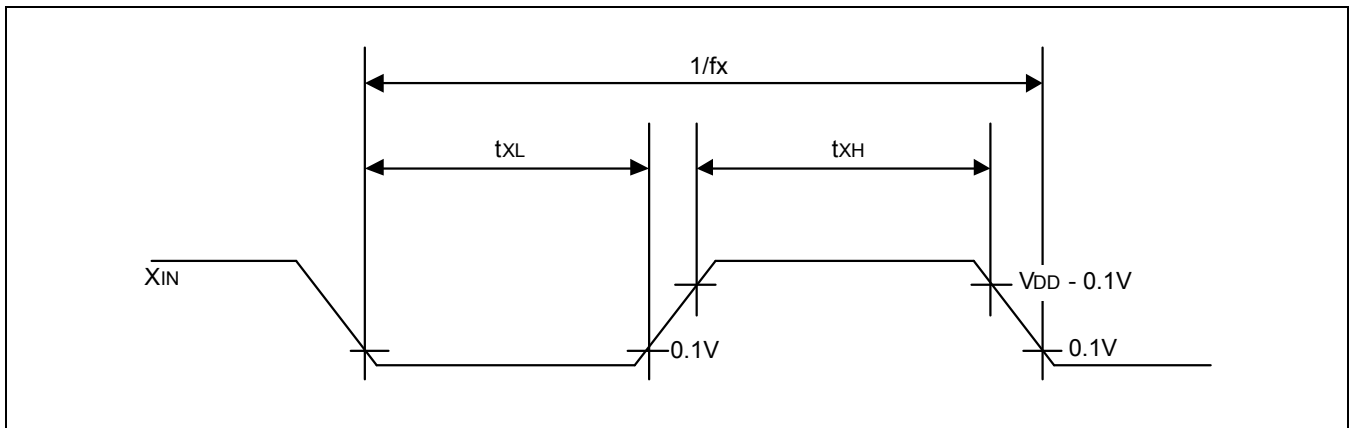
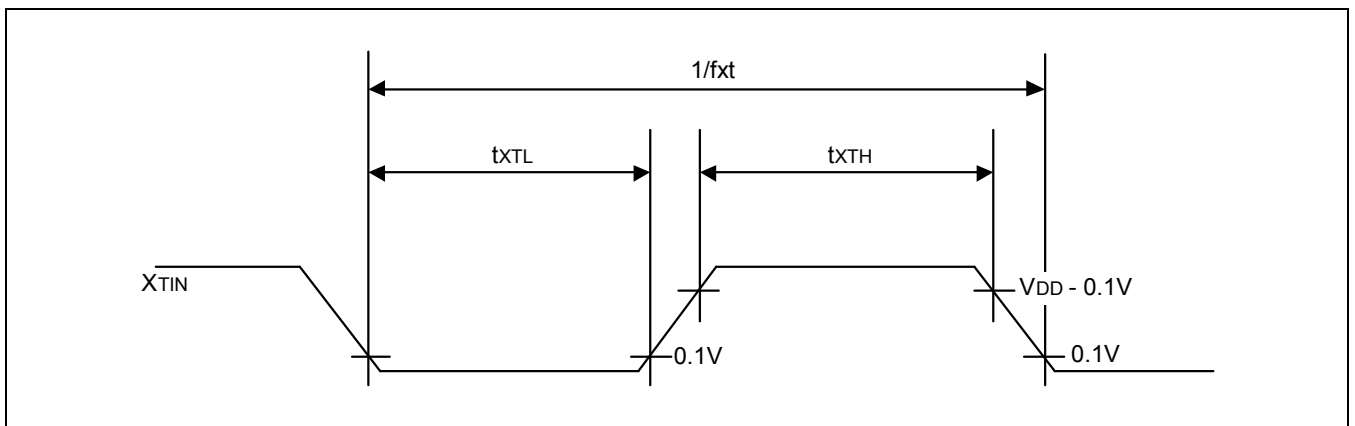
图 20-9  $X_{IN}$ 管脚测得的时钟时序

表 20-14 副振荡稳定时间

 $(T_A = -25^{\circ}\text{C} - +85^{\circ}\text{C}, V_{DD} = 2.0\text{V} - 3.6\text{V})$ 

振荡器	测试条件	最小值	典型值	最大值	单位
石英振荡器	—	—	—	10	s
外部时钟源	$XT_{IN}$ 输入高低电平宽度 ( $t_{XTH}, t_{XTL}$ )	5	—	15	$\mu\text{s}$

图 20-10  $XT_{IN}$ 管脚测得的时钟时序

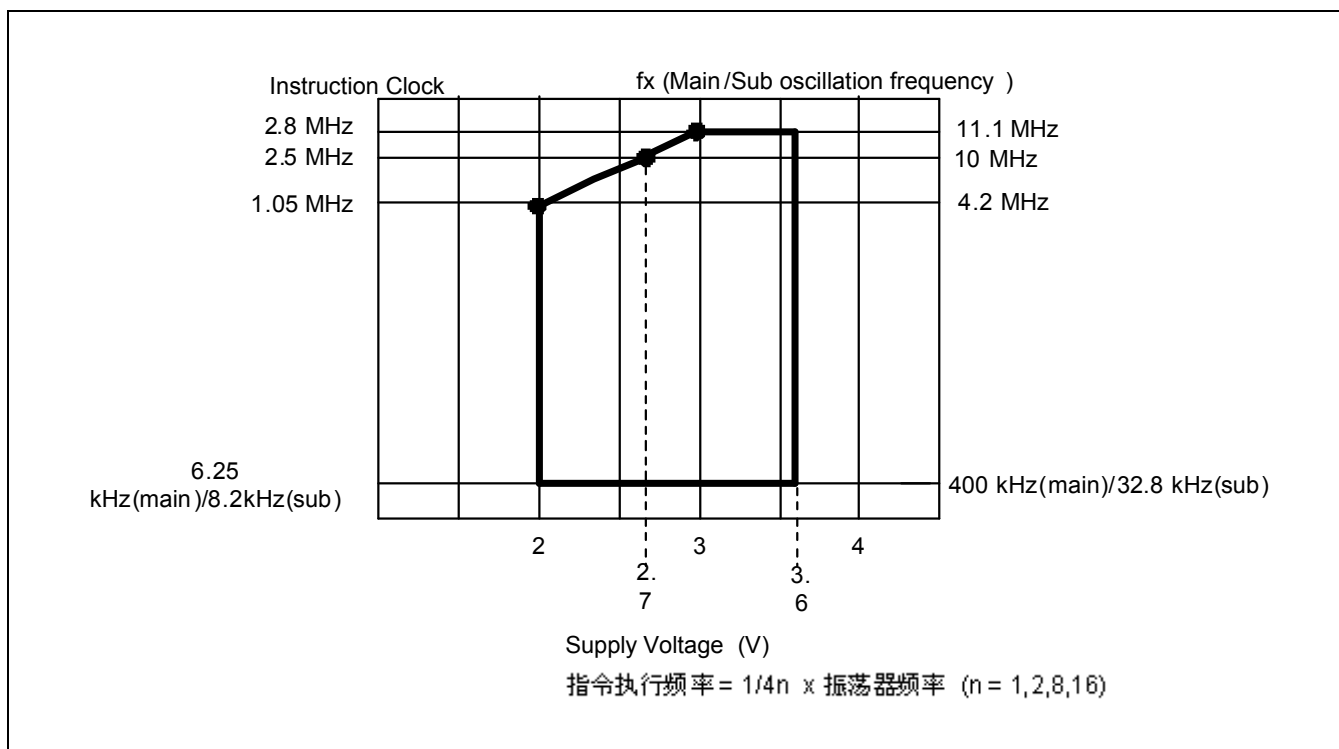


图 20-11 工作电压范围

表 20-15 内部 Flash ROM 电气特性

( $T_A = -25^{\circ}\text{C} - +85^{\circ}\text{C}$ ,  $V_{DD} = 2.2\text{V} - 3.6\text{V}$ )

参数	标号	条件	最小值	典型值	最大值	单位
编程时间 (1)	Ftp	-	30	-	-	$\mu\text{s}$
芯片擦除时间 (2)	Ftp1	-	50	-	-	ms
扇区擦除时间 (3)	Ftp2	-	10	-	-	ms
读频率	Ft <sub>RS</sub>	-	-	25	-	ns
写/擦除次数	FN <sub>WE</sub>	-	-	-	10,000 <sup>(4)</sup>	Times

注释:

1. 编程时间是编程一字节 (8 位) 所需要的时间。
2. 芯片擦除时间是擦除整个 64K 字节所需要的时间。
3. 扇区擦除时间是擦除一个扇区内所有 128 字节所需要的时间。
4. Full-flash((S3F828BX)的最大擦写次数是 10,000 次, half flash(S3F8289X/F8285X)的最大擦写次数是 100 次
5. 芯片擦除只在工具编程模式下可用。

# 21

## 机械尺寸

### 21.1 概述

S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X 现有的封装类型有: 80-pin-QFP/TQFP 封装。

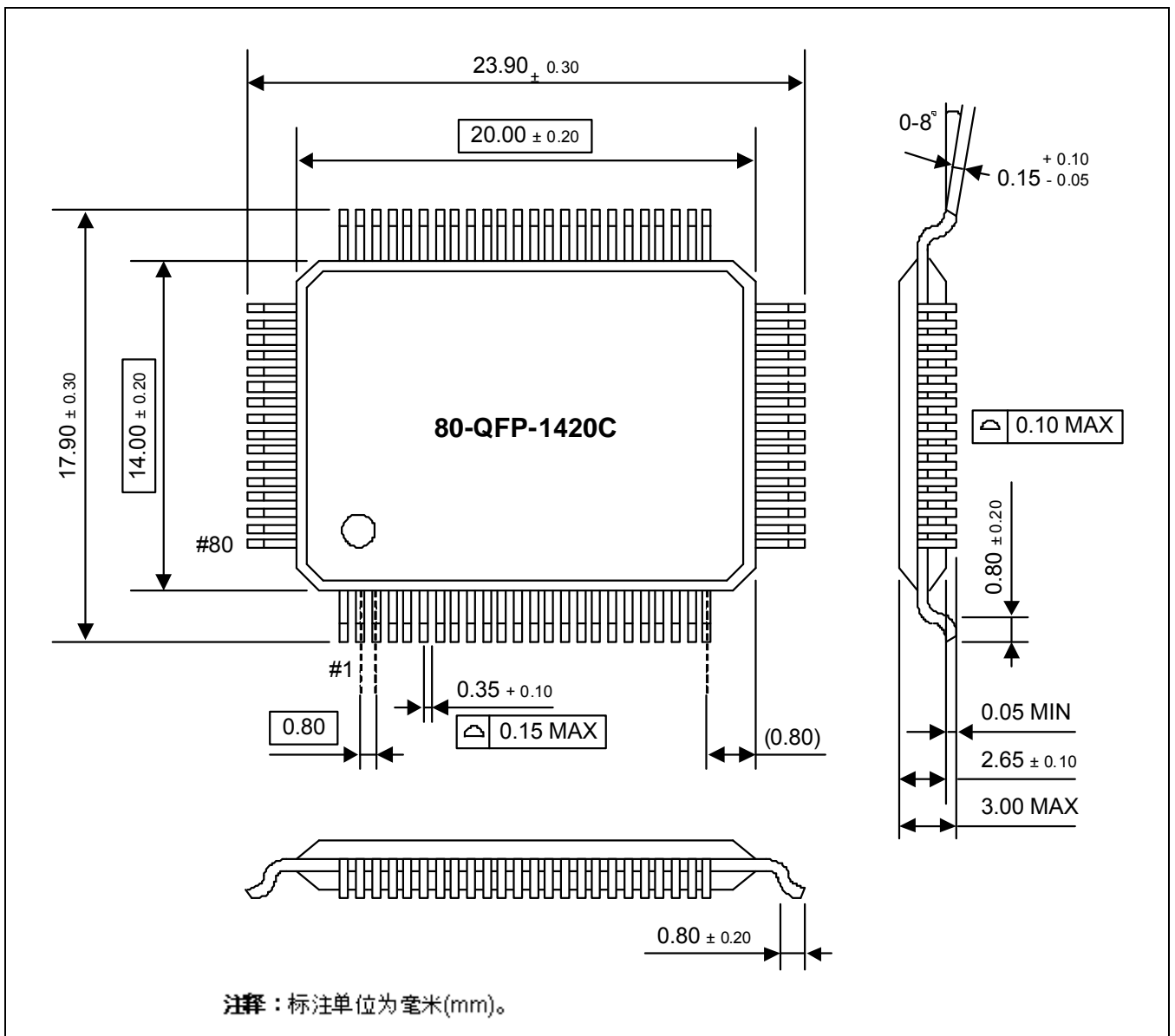


图 21-1 封装尺寸(80-QFP-1420C)

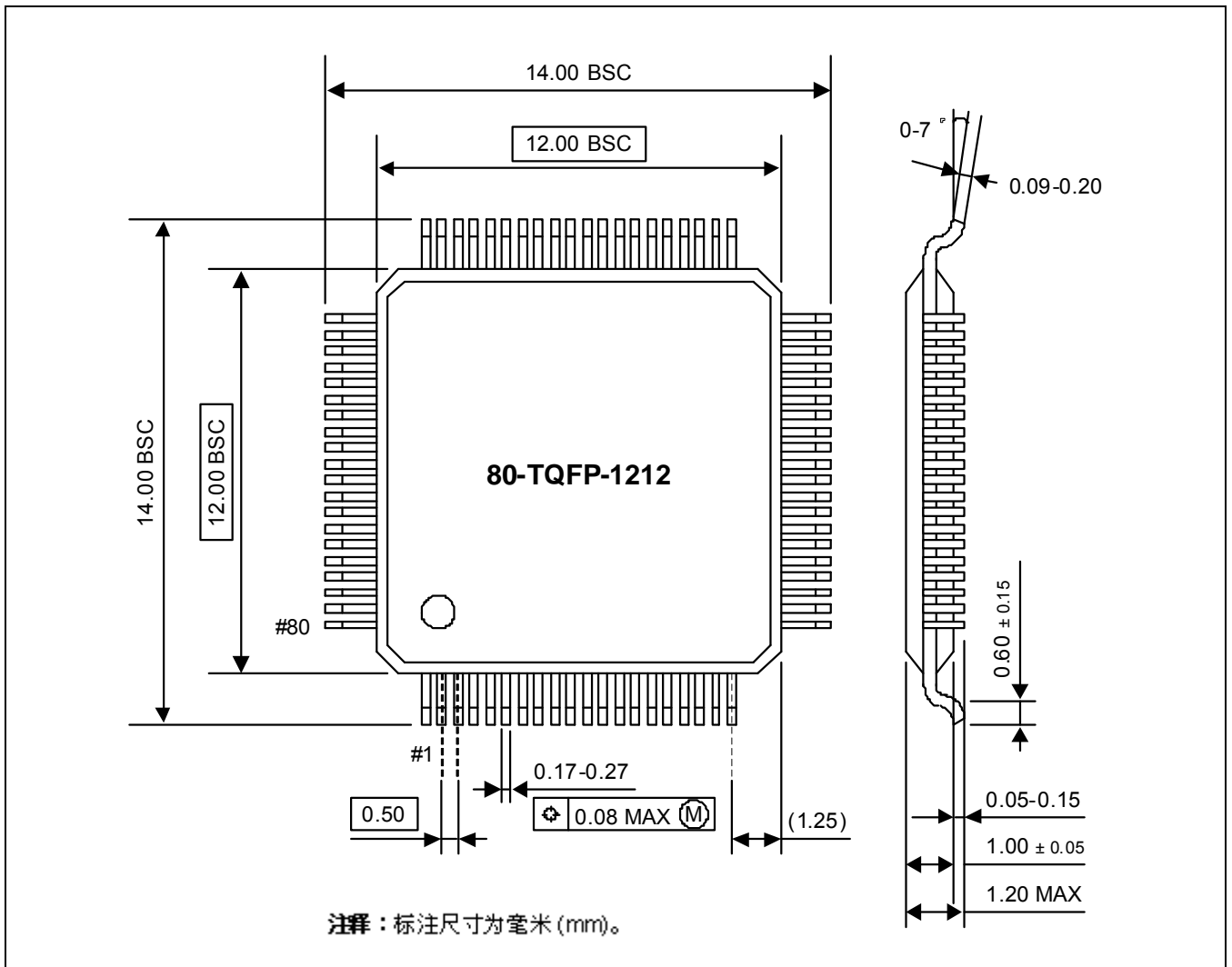


图 21-2 封装尺寸(80-TQFP-1212)

# 22

## S3F828BX/F8289X/F8285X FLASH MCU

### 22.1 概述

S3F828BX/F8289X/F8285X 单片CMOS MCU是S3C828BX/C8289X/C8285X 的Flash MCU版本。它具有片上16K/8K/4K 字节闪存空间，可以通过串行数据格式访问。

S3F8275X/F8278X/F8274X 和S3C8275X/C8278X/C8274X 在功能和管脚上都是完全兼容的。因为编程简单可重复，所以S3F8275X/F8278X/F8274X 非常适合用作S3C8275X/C8278X/C8274X的评估芯片。

**注释：** 本章介绍 Flash MCU 的工具编程模式。如需了解用户编程模式，请参考第 19 章“嵌入式闪存接口”。

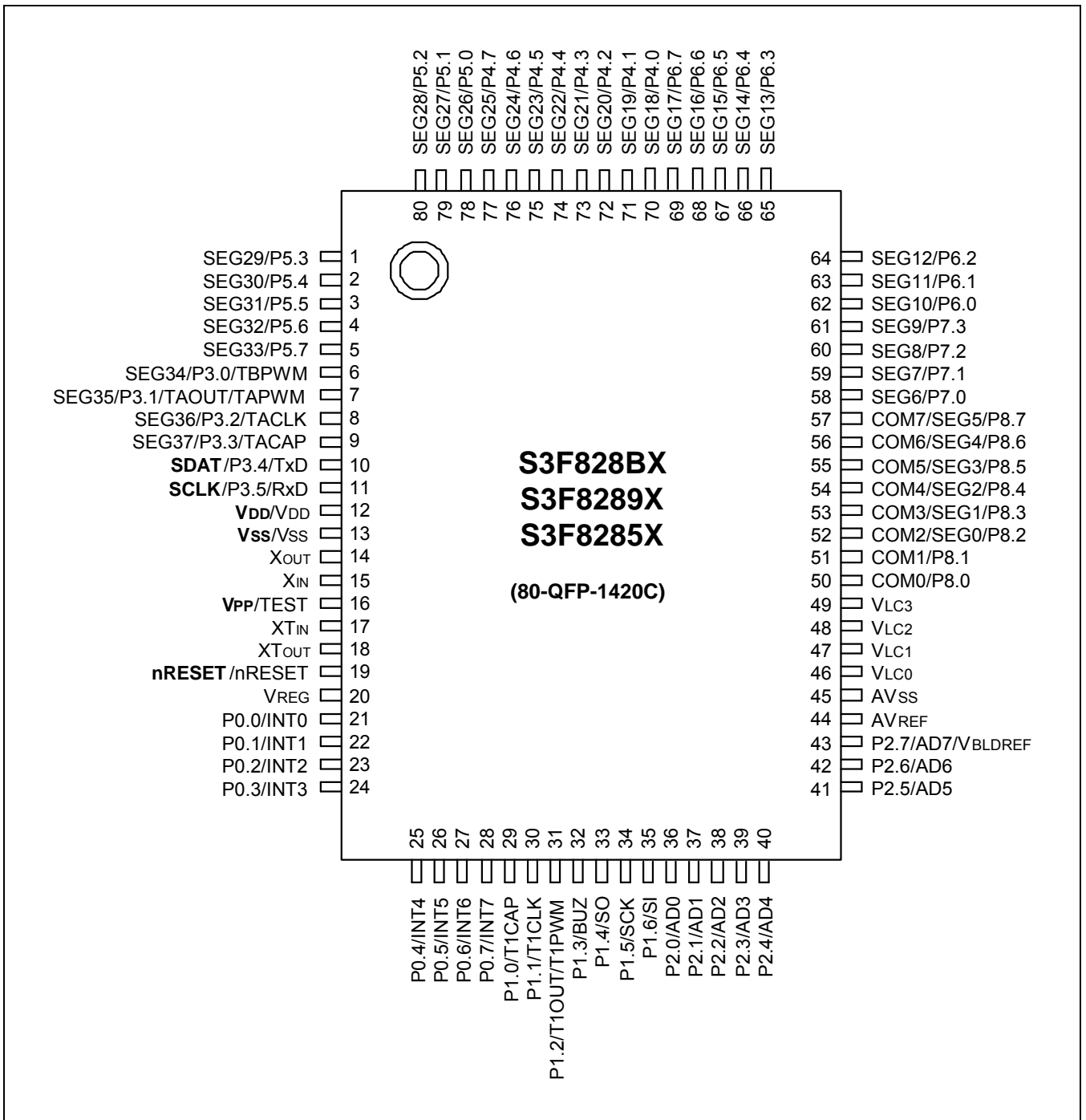


图 22-1 S3F828BX/F8289X/F8285X 管脚分配 (80-QFP-1420C)

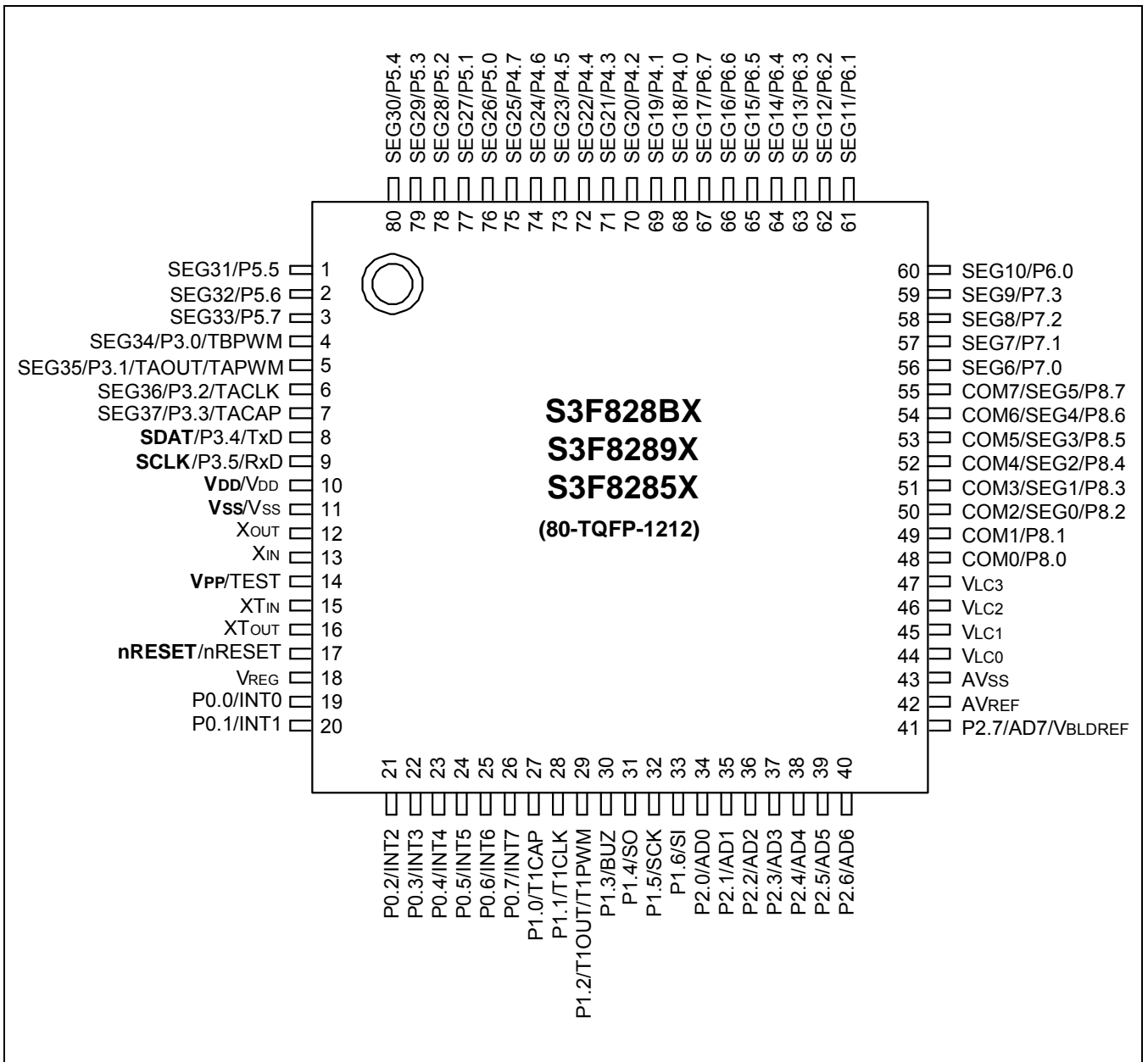


图 22-2 S3F828BX/F8289X/F8285X 管脚分配 (80-TQFP-1212)



表 22-1 闪存读写管脚描述

主芯片 管脚名	编程过程中			
	管脚名称	管脚 No.	I/O	功能
P3.4	SDAT	10(8)	I/O	串行数据管脚（读时为输出脚，写入时为输入脚），管脚可设置为输入和推挽式输出模式。
P3.5	SCLK	11(9)	I/O	串行时钟管脚（仅为输入管脚）。
TEST	V <sub>PP</sub>	16(14)	I	S3F8289X/F8285X: Flash ROM单元读写供电管脚。管脚检测到12.5V时进入Flash写模式，检测到3.3V时进入Flash读模式。 S3F828BX: Flash ROM单元读写供电管脚。因为内部自升压电路，所以无论Flash写或者读模式，都只需要3.3V。所以TEST管脚接到VDD即可。
nRESET	nRESET	19(17)	I	芯片初始化。
V <sub>DD</sub> , V <sub>SS</sub>	V <sub>DD</sub> , V <sub>SS</sub>	12(10) 13(11)	-	逻辑供电管脚

## 注释:

1. 括号内是 80-TQFP-1212 封装形式的芯片管脚 No。
2. V<sub>PP</sub> (Test) 管脚必须连到 V<sub>DD</sub> (S3F828BX only)。

表 22-2 S3F828BX/F8289X/F8285X 和 S3C828BX/C8289X/C8285X特性比较

特性	S3F828BX/9X/5X	S3C828BX/9X/5X
程序存储空间	64K/32K/16K-byte Flash ROM	64K/32K/16K-byte 掩模 ROM
工作电压 (V <sub>DD</sub> )	2.0 V to 3.6V	2.0V to 3.6V
Flash ROM 编程模式	V <sub>DD</sub> = 3.3V, V <sub>PP</sub> (TEST) = 12.5V	-
管脚配置(封装)	80-QFP, 80-TQFP	80-QFP, 80-TQFP
Flash ROM可编程性	用户可多次编程	工厂烧程序

### 22.1.1 编程工作模式特性

当在S3C828BX/C8289X/C8285X 的 $V_{PP}(\text{TEST})$ 管脚上加上12.5V电压时，Flash ROM即进入编程模式。

编程工作模式（读，写，读保护）由下表控制信号选择。

表 22-3 编程工作模式选择

VDD	VPP(TEST)	REG/nMEM	地址 (A15-A0)	R/W	模式
3.3 V	3.3V	0	0000H	1	Flash ROM 读
	12.5V	0	0000H	0	Flash ROM 编程(写)
	12.5V	0	0000H	1	Flash ROM 校验
	12.5V	1	0E3FH	0	Flash ROM 读保护

**注释:**

1. S3F828BX 中， $V_{PP}(\text{Test})$  管脚必须连到  $V_{DD}$ 。
2. "0" 表示逻辑低电平； "1" 代表逻辑高电平。

# 23

## 开发工具

### 23.1 概述

三星提供了一套强大易用的开发工具，该开发工具由一个主系统，一套调试工具和相应的支持软件组成。所谓主系统，即任何一台使用 Win95/98/2000/XP 操作系统的标准电脑。

成熟的调试工具包含了硬件和软件：一个强大的电路仿真器，SMDS2+ 或OPENice，支持所有的 S3C7-, S3C9- 和 S3C8- MCU 家族。SMDS2+为SMDS2的最新版本。三星同时可提供各种支持仿真器的软件，包括调试器、汇编编译器和仿真器安装设置程序。

#### 23.1.1 SHINE

三星的在线仿真器主机接口，SHINE，是一个支持多窗口的SMDS2+调试器。SHINE可以提供上拉和弹出菜单，支持鼠标操作，功能热键，和基于文本的超链接帮助。它具有先进的，多窗口的用户界面，强调用户体验。每一个窗口都可调节大小、移动，高亮、增加或完全移除。

#### 23.1.2 SAMA 编译器

三星的可调节微控制器汇编器，SAMA，是一个通用的汇编器，可以产生标准的hex格式的目标文件。汇编程序代码包括程序数据目标文件和所需的SMDS程序控制数据。汇编程序时，SAMA需要一个源文件和一个包括产品特定信息的辅助定义文件(DEF)。

#### 23.1.3 SASM88

SASM88是一个可重定位的适用于三星S3C8系列微控制器的汇编器。SASM88将一个包括汇编语言的源文件汇编后生成源代码，目标代码及注释。SASM88支持宏和条件编译。它在MS-DOS模式下运行。

它仅产生的可重定位的目标文件。因此用户需要链接目标文件。一个目标文件可以和另一个目标文件进行链接后装载到存储器中。

#### 23.1.4 HEX2ROM

HEX2ROM将由汇编器产生的hex格式的文件转换生成可执行的存储器代码。ROM代码需要烧写到掩模ROM中。在生产ROM 代码时，ROM中未使用的字节用"FF"来填充以达到适合目标芯片的最大的存储器大小。

#### 23.1.5 目标板

S3C8 - 系列 MCU 调试时需要目标板，该板上提供了所有需要的目标系统连线和调试接口。

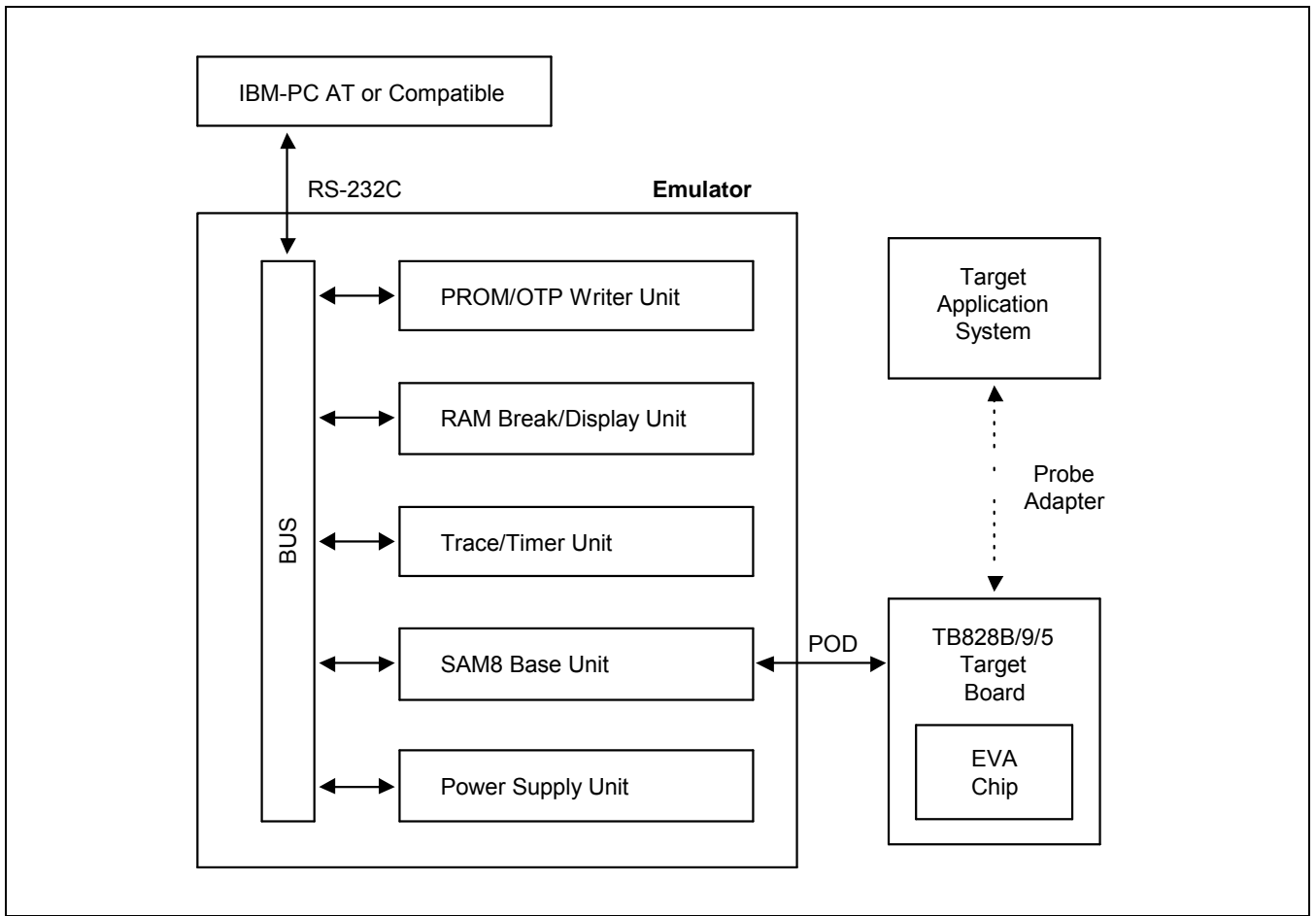


图 23-1 开发系统配置

23.1.6 TB828B/9/5 目标板

TB828B/9/5目标板可用于 S3C828BX/F828BX/C8289X/F8289X/C8285X/F8285X MCU的开发。TB828B/9/5目标板与仿真器 (SMDS2+, SK-1000, OPENice-i500) 相连时作为目标CPU工作。

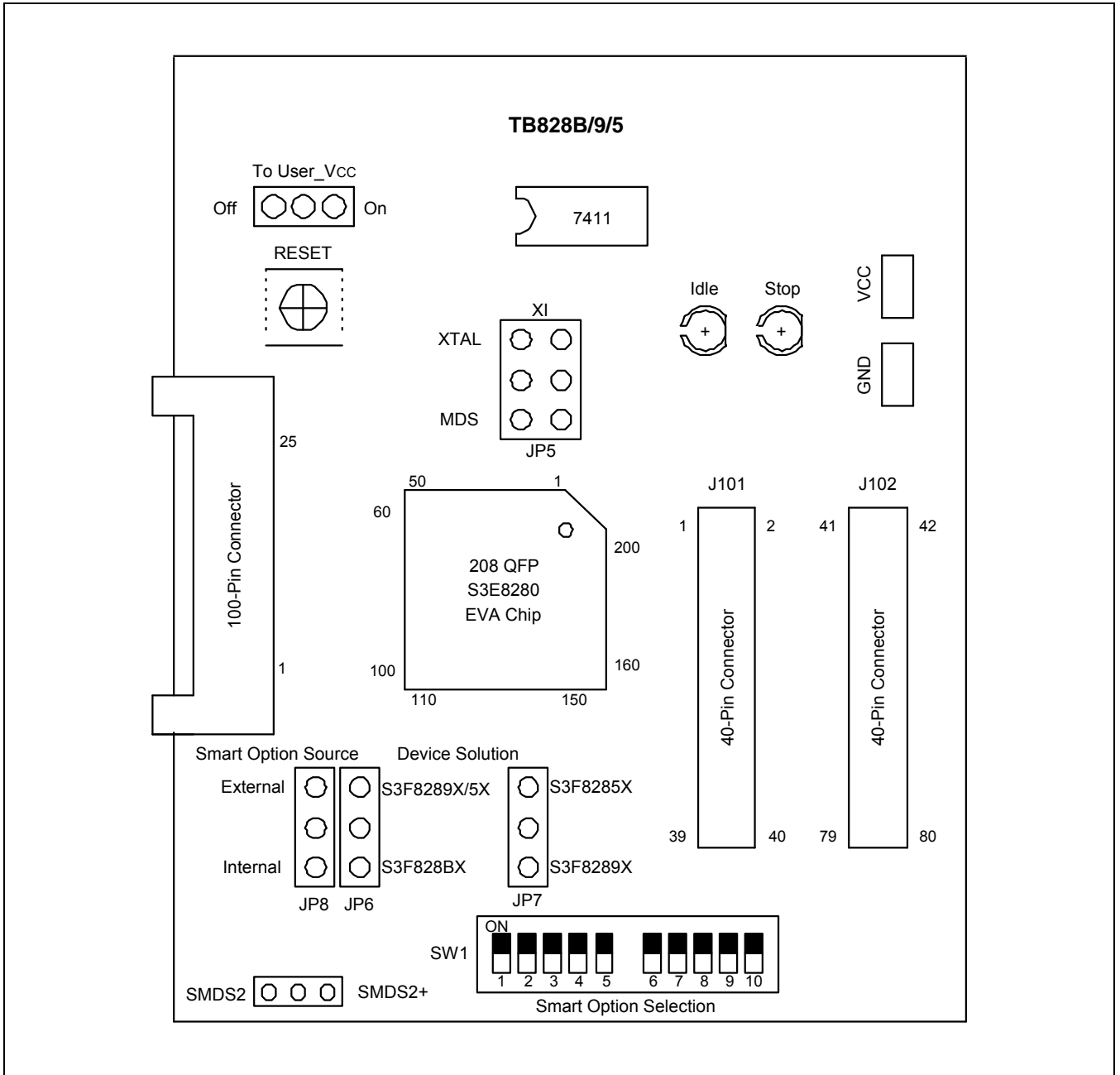

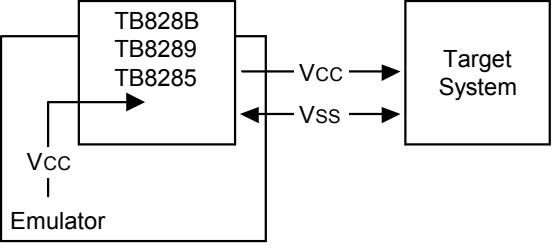

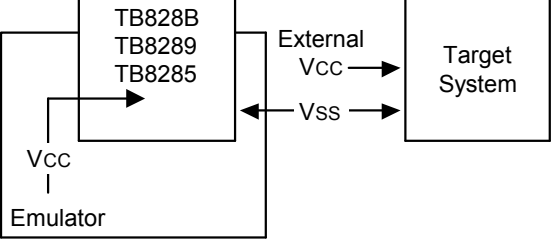


图 23-2 TB828B/9/5 目标板配置

表 23-1 TB828B/9/5 电源选择设定

"To User_Vcc" 设置	工作模式	注释
<p>To User_Vcc Off  On</p>		<p>仿真器提供VCC 给目标板 (评估芯片)和目标系统。</p>
<p>To User_Vcc Off  On</p>		<p>仿真器提供VCC 给目标板 (评估芯片) 目标系统需要有自己的电源</p>

注释: 下面的标记 "To User\_Vcc" 表示两点之间短接:



表 23-2 TB828B/9/5 主时钟选择控制

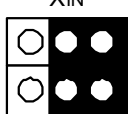
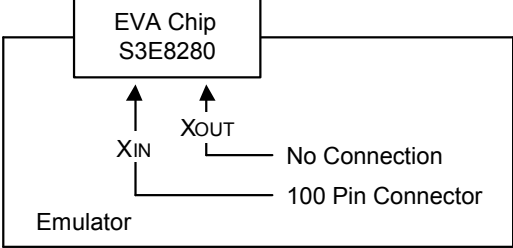
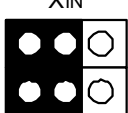
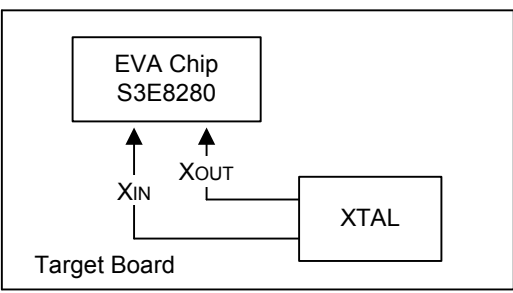

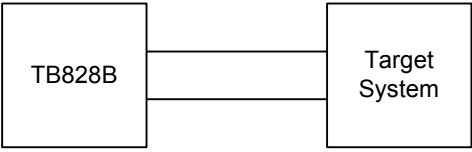


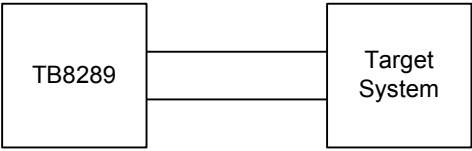


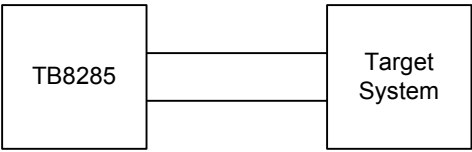
主时钟设置	工作模式	注释
<p>XIN XTAL  MDS</p>		<p>当把目标板链接到仿真器, 设定XI 切换到 "MDS"</p>
<p>XIN XTAL  MDS</p>		<p>设定 XI 切换到 "XTAL",使用目标板上自身的时钟源</p>


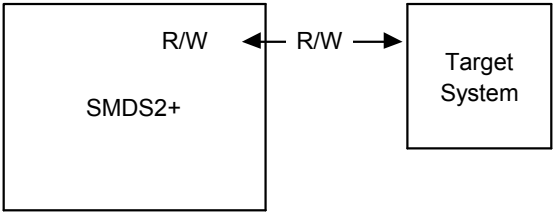
表 23-3 TB828B/9/5的芯片选择

"Device Selection" 设置	工作模式	注释
Device Selection 8289X/5X  828BX		选择 TB828B
Device Selection 8289X/5X  828BX 8285X  8289X		选择 TB8289
Device Selection 8289X/5X  828BX 8285X  8289X		选择 TB8285

### 23.1.7 SMDS2+ 选择 (SAM8)

为了实现SMDS2+的向程序存储器中写入数据的功能，目标板连接到SMDS2+后，需要对开关进行如下的设置。否则，此功能不可用。

表 23-4 SMDS2+ 工具选择设置

"SMDS2+" 设置	工作模式
SMDS2  SMDS2+	

#### 23.1.7.1 IDLE LED

黄色的 LED 亮时表示 评估芯片(S3E8280)进入了idle 模式。

#### 23.1.7.2 STOP LED

红色的 LED 亮表示评估芯片(S3E8280) 进入了stop 模式。

表 23-5 TB828B/9/5 Smart Option源设置


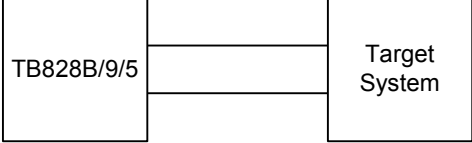

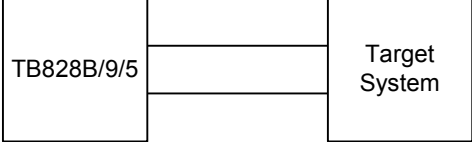
"Smart Option Source" 设置	工作模式	注释
<p>Select Smart Option Source</p> <p>Internal  External</p>		<p>Smart Option 由外部开关来选择(SW1)</p>
<p>Select Smart Option Source</p> <p>Internal  External</p>		<p>Smart Option 由芯片内部的smart option 区域决定 (ROM 的003EH-003FH). 但是此选择不可用。</p>

表 23-6 TB828B/9/5 的Smart Option设置

"Smart Option" 设置	注释
	<p>Smart Option 由此开关的状态确定。SW1.3-SW1.1对应003EH.2-.0. SW1.8-SW1.6对应003EH.7-.5. SW1.9对应003FH.0. The SW1.5-1.4没有连接. SW1.10 未使用。</p>



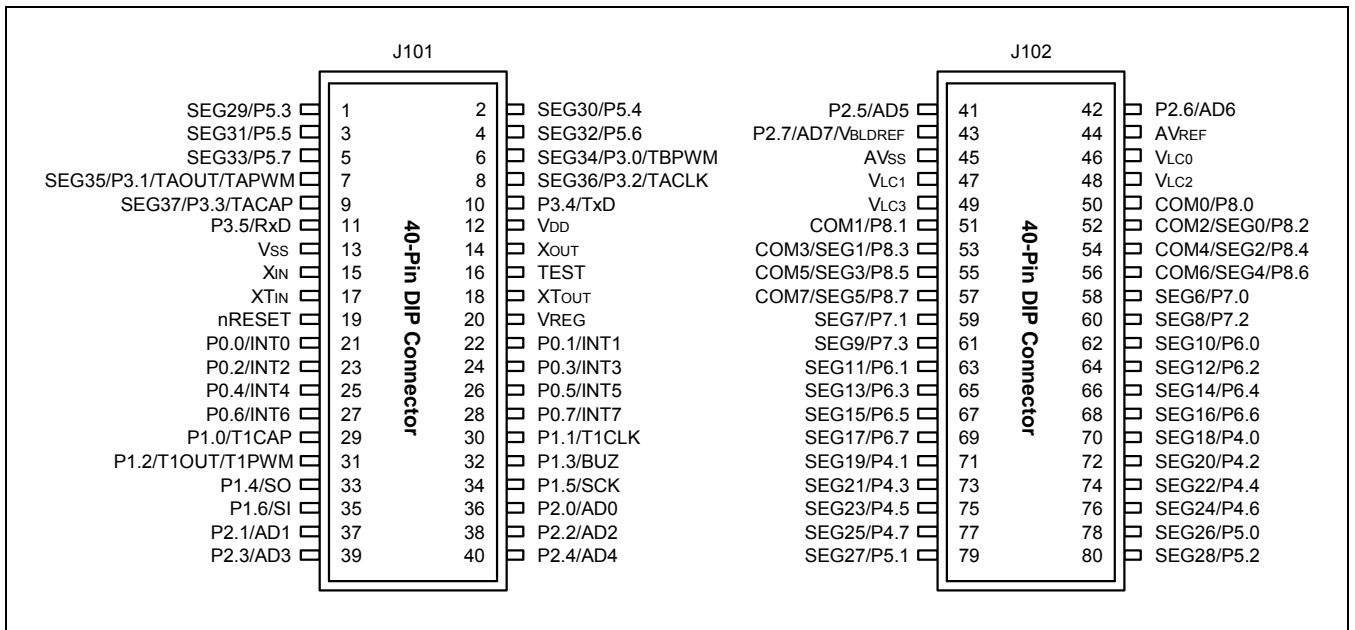


图 23-3 TB828B/9/5的40脚接口(J101, J102)

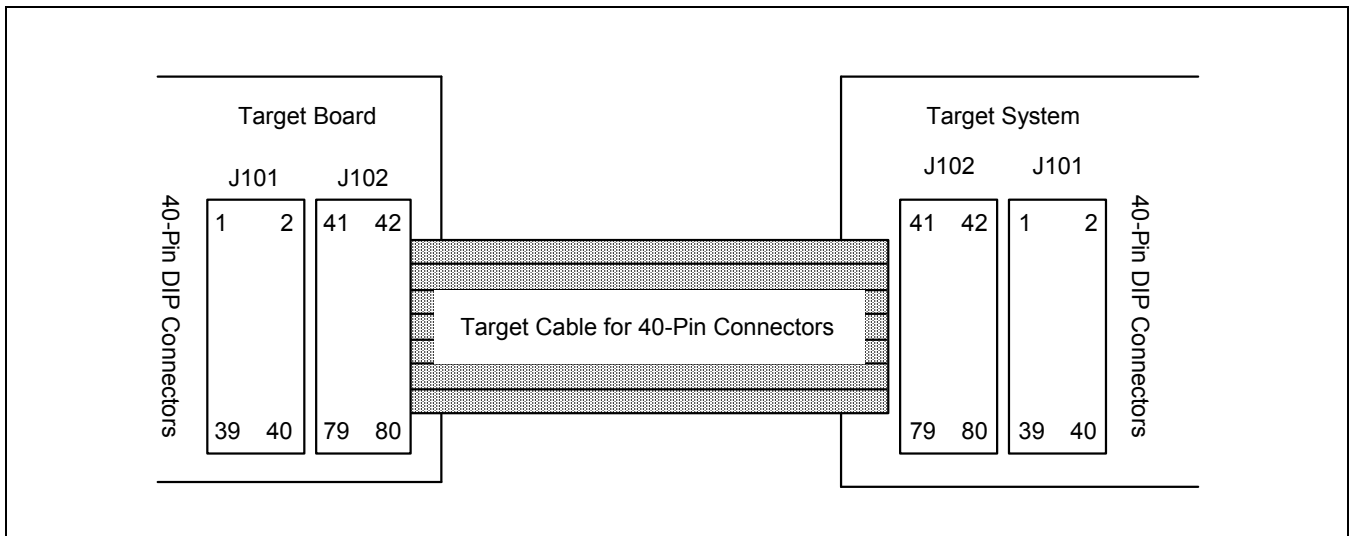


图 23-4 专为80-QFP 封装的S3E8280的数据线